INTERACTIVE GRAPHICAL INTERFACE FOR PRINTED GLYCAN ARRAY DATA ANALYSIS

_____

A Thesis

Presented to the

Faculty of

San Diego State University

_____

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

in

Computer Science

_____

by

William Anderson King

Fall 2011

SAN DIEGO STATE UNIVERSITY

The Undersigned Faculty Committee Approves the

Thesis of William Anderson King:

Interactive Graphical Interface for Printed Glycan Array Data Analysis

Marko I. Vuskovic, Chair
Department of Computer Science

Marie A. Roch
Department of Computer Science

Christopher P. Paolini
Computational Science Research Center

11/14/2011

Approval Date

Copyright © 2011
by
William Anderson King
All Rights Reserved

DEDICATION

I would like to dedicate this thesis to my father, William King, and mother, Carol King, who have always encouraged me to learn. They have both supported me throughout my entire educational career, including all of the times I decided to procrastinate. I could not have finished this project without their support.

I would also like to dedicate this thesis to my girlfriend, Jaren Dollard. She has been incredibly supportive during the process, making sure that I was well rested, nourished, and happy.

ABSTRACT OF THE THESIS

Interactive Graphical Interface for Printed Glycan Array Data Analysis
by
William Anderson King
Master of Science in Computer Science
San Diego State University, 2011

This thesis presents a specification, implementation, and description of the GlycoAnalyzer application; a Bioinformatics graphical user interface-based tool which is particularly tuned for analyzing glycan-based data obtained from printed glycan arrays (PGA). PGAs are micro arrays based on new high-throughput technology, similar to protein and DNA arrays, but contain a library of glycans covalently attached to the array glass instead of proteins or DNAs. Such arrays are used to measure activity of the immune system in order to perform screening of the general population, early detection of cancerous and viral diseases, and diagnosis and prognosis of these diseases by observing the level of anti-glycan antibodies present in human blood.

The GlycoAnalyzer performs preprocessing of raw data obtained from PGAs and performs down-stream analysis, which includes feature selection, classification, and visualization of data. All aspects of the PGAs and processing of PGA data, as well as implementation of the GlycoAnalyzer are described and a working example is presented which contains a mesothelioma assay that consists of a control group of 65 subjects exposed to asbestos and 50 patients with malignant mesothelioma. Future plans for a mobile version of the GlycoAnalyzer are also discussed.

TABLE OF CONTENTS

PAGE

ABSTRACT v
LIST OF TABLES ix
LIST OF FIGURES x
ACKNOWLEDGEMENTS xiii
CHAPTER
1 INTRODUCTION 1
2 PRINTED GLYCAN ARRAYS (PGA) 3
3 DATA PROCESSING AND ANALYSIS USED IN GRAPHICAL DATA ANALYSIS 6
3.1 Background 6
3.2 Data Preprocessing 8
3.3. Measuring the Goodness of Discrimination 11
3.3.1 Student and Wilcoxon Statistic 11
3.3.2 Support Vector Machines 14
3.3.3 Receiver Operating Characteristic (ROC) Curve 14
3.3.4 Specificity and Sensitivity 16
3.3.5 Area Under the ROC Curve 18
3.3.6 Adjusted ROC Curve 18
3.4 Feature Selection 20
3.4.1 Univariate Methods 20
3.4.1.1 Student Ranking 21
3.4.1.2 Wilcoxon Ranking 21
3.4.2 Multivariate Methods 22
3.4.2.1 Fisher Linear Discriminant 23
3.4.2.2 Backward Stepwise Feature Selection (RFE and GUYON) 23
3.4.2.3 Forward Stepwise Feature Selection (RFA and RFA_L) 24
3.5 Classification 24

3.6 Data Visualization....24
3.6.1 ImmunoRuler Plots....25
3.6.1.1 ImmunoRuler Plot with Quartile Regions....28
3.6.1.2 Simple ImmunoRuler Plot....28
3.6.2 Probability Density Functions (PDF)....30
3.6.3 Receiver Operating Characteristic (ROC) Curves....32
4 FUNCTIONALITY OF THE GLYCOANALYZER....34
4.1 Installing the GlycoAnalyzer Application....34
4.2 Launching and Closing the GlycoAnalyzer Application....35
4.3 Application Button Color Codes....36
4.4 Incorrect User Operations and Errors....38
4.5 Main Window, Data Input Controls Section....40
4.6 Main Window, Preprocessing Controls Section....42
4.7 Main Window, Feature Selection and Projection Controls Section....44
4.8 Main Window, Plotting Controls Section....46
4.9 Main Window, Status and Error Controls Section....49
4.10 Preprocessing Window....50
4.11 Output Window....51
4.12 Plot Window....52
5 IMPLEMENTATION OF THE GLYCOANALYZER IN THE MATLAB GUI ENVIRONMENT....54
5.1 General Description....55
5.2 Support Functions....58
5.3 Structure of the MATLAB GUI Run-Time System....59
5.4 Compiling MATLAB Code and Building the Stand-Alone Application....61
5.4.1 Locating and Setting-up the Installed and Supported Compilers....61
5.4.2 Deploying the GlycoAnalyzer to End-Users....62
5.4.2.1 Building a New GlycoAnalyzer Deployment Project....62
5.4.2.2 Building an Existing GlycoAnalyzer Deployment Project....63
5.4.2.3 Packaging the GlycoAnalyzer Application for Deployment....64
5.4.2.4 Deploying the GlycoAnalyzer Application to End-Users....65
5.5 General Application Update....65

5.5.1 Updating Existing Functions in the GlycoAnalyzer Application 66
5.5.2 Adding New Files to the GlycoAnalyzer Application 66
5.5.3 Deleting Files from the GlycoAnalyzer Application 67
5.5.4 Adding Components to the GlycoAnalyzer Application 67
5.5.5 Deleting Components from the GlycoAnalyzer Application.....68
5.5.6 Adding Auxiliary Windows to the GlycoAnalyzer Application.....68
5.5.7 Deleting Auxiliary Windows from the GlycoAnalyzer Application.....70
5.6 Implementation Issues 70
6 RESULTS 74
7 MOBILE GLYCOANALYZER.....87
8 CONCLUSION.....90
REFERENCES 92
APPENDIX
A GLYCOANALYZER COMPONENT DESCRIPTIONS 96
B GLYCOANALYZER GLOBAL VARIABLE DESCRIPTIONS.....108
C GLYCOANALYZER FILES AND FUNCTIONS 112

LIST OF TABLES

PAGE

Table 3.1. ROC Contingency Table.....16
Table 5.1. Files Created During Compilation.....60

LIST OF FIGURES

PAGE

Figure 2.1. Sample of individual patient arrays. 3
Figure 2.2. Binding of the human antibodies and goat anti-human antibodies to the glycan structures on the PGA.. 4
Figure 2.3. Image from a developed PGA sub-array. 5
Figure 2.4. Steps in preparing and processing the PGA and the steps involved in the data analysis. 5
Figure 3.1. Graphical representation of the raw dataset packed in structure *D*. 8
Figure 3.2. Graphical comparison between the hypotheses H0 and H1. 13
Figure 3.3. Graphical representation of the SVM concept. 15
Figure 3.4. Hypothetical plot of a specific feature. 16
Figure 3.5. Sample ROC diagram for the mesothelioma assay displaying the adjusted ROC curve. 19
Figure 3.6. Sample ImmunoRuler plot.. 26
Figure 3.7. Sample ImmunoRuler plot, IR new. 29
Figure 3.8. Sample ImmunoRuler plot, IR. 30
Figure 3.9. Sample individual PDF plots. 31
Figure 3.10. Sample combined PDF plot. 31
Figure 3.11. Sample individual ROC plot. 32
Figure 3.12. Sample combined POC plot. 33
Figure 4.1. File structure of GlycAnalyzer_pkg.exe and file creation flow from deploytool. 35
Figure 4.2. GlycoAnalyzer Close dialog box. 36
Figure 4.3. Red Browse button before the training data is loaded. 37
Figure 4.4. Red Browse button after the training data is loaded. 37
Figure 4.5. Orange user error notification after an incorrect sequence of events. 39
Figure 4.6. Orange user error notification after an incorrect value is entered in an editable textbox. 39
Figure 4.7. Orange “?” Button after a programming error has occurred. 39
Figure 4.8. Generate Error dialog box. 40

Figure 4.9. GlycoAnalyzer Data Input Controls section.....40
Figure 4.10. GlycoAnalyzer Delete File dialog box.....41
Figure 4.11. Preprocessing Controls Section with initial values.....42
Figure 4.12. Preprocessing Controls after preprocessing is complete.....43
Figure 4.13. Feature Selection and Projection Controls before preprocessing.....44
Figure 4.14. Feature Selection and Projection Controls after preprocessing.....45
Figure 4.15. Plotting Controls allowing the user to select the plot type.....47
Figure 4.16. Plotting Controls for modifying and displaying the plot.....47
Figure 4.17. Sample IR new plot once plotting is complete.....48
Figure 4.18. Status and Error Controls section.....49
Figure 4.19. Preprocessing window after preprocessing is complete.....50
Figure 4.20. Output window after feature selection and projection is complete.....51
Figure 4.21. Plot window with an example IR plot after plotting is complete.....52
Figure 5.1. Development flow of the GlycoAnalyzer.....54
Figure 5.2. User installation and operational flow of the GlycoAnalyzer.....55
Figure 5.3. Blank MATLAB GUI Layout Editor window.....56
Figure 5.4. Property Inspector for the Feature Selection pop-up menu.....58
Figure 5.5. Diagram of GlycoAnalyzer function structure.....59
Figure 5.6. Function: My_close.....72
Figure 5.7. Function: My_error.....72
Figure 6.1. Open GlycoAnalyzer application in an initial state.....74
Figure 6.2. Training Data Search dialog box.....75
Figure 6.3. Data Input Controls section after the training data is loaded.....75
Figure 6.4. Data Labels Search dialog box.....76
Figure 6.5. Data Input and Preprocessing Controls sections after the data labels have been loaded.....76
Figure 6.6. Preprocessing and Feature Selection/Projection Controls sections after preprocessing is completed.....77
Figure 6.7. Preprocessing window after preprocessing is complete.....78
Figure 6.8. Checked checkboxes in the Feature Selection/Projection Controls section.....79
Figure 6.9. Feature Selection/Projection and Plotting Controls sections after feature selection and projection are completed.....79
Figure 6.10. Output window after feature selection and projection are complete.....80

Figure 6.11. Completed ImmunoRuler plot. 81
Figure 6.12. Plot window after completed ImmunoRuler plot. 81
Figure 6.13. Replotted ImmunoRuler after a change in the threshold height. 82
Figure 6.14. ImmunoRuler tool tip. 83
Figure 6.15. Individual ROC plots for six top features 84
Figure 6.16. Combined ROC plot for six top features. 84
Figure 6.17. Individual PDF plot for six top features. 85
Figure 6.18. Combined PDF plot for six top features. 85
Figure 7.1. Data Input Controls running on iOS 88
Figure 7.2. Preprocessing Controls running on iOS. 88
Figure 7.3. Feature Selection and Projection Controls running on iOS 89

ACKNOWLEDGEMENTS

I would like to thank Professor Marko Vuskovic for his assistance and guidance throughout the GlycoAnalyzer project and for sharing his programs that are incorporated in the GlycoAnalyzer Engine.

I would also like to thank Dr. Margaret Huflejt from New York University School of Medicine for providing the PGA data that was used for testing the GlycoAnalyzer.

Finally, I would like to thank Dr. Marie Roch and Dr. Christopher Paolini for being members of my defense board and for reviewing my thesis.

CHAPTER 1

INTRODUCTION

The American Cancer society recommends specific screening guidelines to assist in the early detection of cancer. These screening guidelines help doctors detect cancers in patients. Early detection is incredibly important because it increases the success-rate of any of the current forms of cancer treatment, including; surgery, radiation, and chemo-therapy. While detecting existing cancer in an early state is very desirable, detecting that cancer before it even exhibits symptoms is even more ideal [1].

While traditional tests like mammograms and colonoscopies have been used to detect cancer, over the past 20 years, different types of biomarkers have been discovered and tested for their reliability in screening for early stage cancer. Two of the major biomarker platforms, include protein biomarkers [2-4] and nucleic acid biomarkers [5, 6]. While research has shown major breakthroughs in cancer detection using these two biomarker platforms, there are drawbacks to each, including (1) expense of the technology, (2) amount of time required for each procedure, (3) narrow targeting of tests for each specific type of cancer (4) variability of patient tissue samples, (5) degrading of tissue samples between the sampling and testing phases, (6) small size of tissue samples on the microarray chip [7].

In the last half-decade, a new biomarker based on printed glycan arrays (PGA) has been gaining in popularity [8]. This paper deals mainly with the development of the GlycoAnalyzer application, a graphical user interface (GUI) created with Mathworks MATLAB, that takes the patient data gathered from PGAs and allows researchers to conduct data preprocessing, feature selection, and projection of data and to graph the results in several different ways. During the past few years, Dr. Marko Vuskovic and his associates have created specific MATLAB functions to analyze and plot the vast amounts of patient data gathered from PGAs. Traditionally, a researcher would use the MATLAB Command Window to load the PGA data and call individual functions or groups of functions required to process and graph the data. While this is an easy task for someone who understands how each file is called, most people without command line experience would probably have a

hard time finding and calling each function properly. Dr. Vuskovic realized that a dedicated GUI that automatically calls the correct function made much more sense for most users unfamiliar with MATLAB.

The GlycoAnalyzer application was developed so that researchers could load patient data gathered from PGAs and stored in a MATLAB specific file, conduct preprocessing, feature selection, and projection of the data, and plot the data to analyze the results from a single user interface. The application can be installed on any PC running Microsoft Windows XP, Vista, or 7 and doesn't require the installation of MATLAB on each individual workstation. Rather than having to know how to use a command line, a user can use the GlycoAnalyzer standard user interface components to load, manipulate, and plot the data.

The purpose of this paper is to document the development and use of the GlycoAnalyzer application in processing the data contained on PGAs. Chapter 2 describes how PGAs work and are prepared and the basic principles behind measuring the levels of human antibodies against the glycans printed on each PGA. Chapter 3 details the principles used during data preprocessing feature selection, projection, and data visualization in the GlycoAnalyzer application. Chapter 4 describes the functionality of the GlycoAnalyzer application and provides a user manual detailing each control in the GUI. Finally, chapter 5 details the implementation of the GlycoAnalyzer in the MATLAB GUI environment.

CHAPTER 2

PRINTED GLYCAN ARRAYS (PGA)

A printed glycan array (PGA) is a glass array on which glycan structures, or complex carbohydrates, are deposited. The surface of the PGA is chemically reactive and allows glycan structures to be attached using covalent bonds during the printing process. The glycan library is printed at two different concentrations (10 and 50 μM), splitting the 16-subarrays of the PGA into two distinct groups of 8 sub-arrays. In total, each sub-array contains a total of 211 glycans with the remainder of the array elements containing biotin spots used as a print control. Each patient data is placed on a unique PGA as in Figure 2.1.

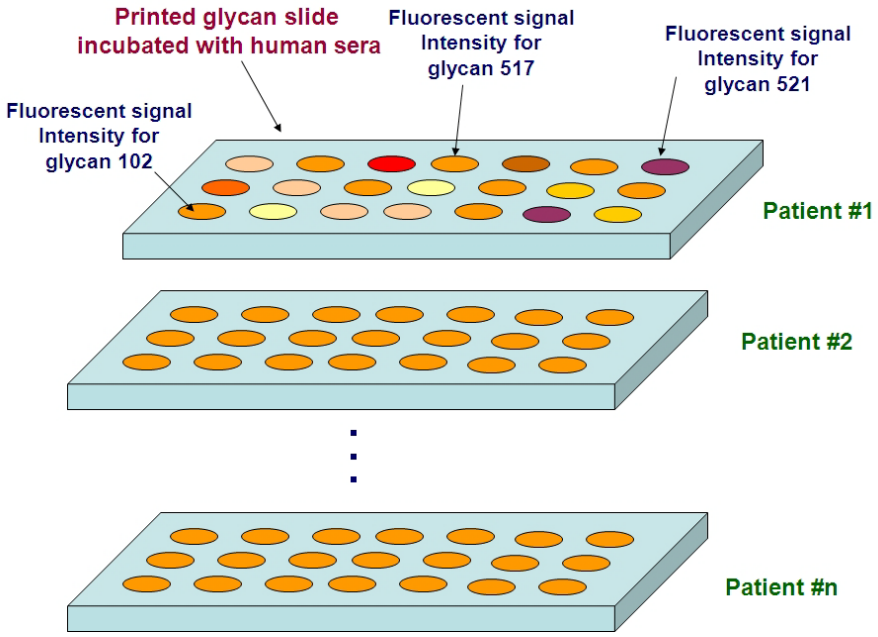
Printed glycan slide incubated with human sera
Fluorescent signal Intensity for glycan 517
Fluorescent signal Intensity for glycan 521
Fluorescent signal Intensity for glycan 102
Patient #1
Patient #2
Patient #n

Figure 2.1. Sample of individual patient arrays. Image property of author given via email by Dr. Marko I. Vuskovic.

Measuring the amount of anti-glycan antibodies that are attached to the individual glycans printed on the PGA is detailed in [9]. An illustration of the binding is found in Figure 2.2. The PGA is first bathed in the patient's serum. This allows the antibodies contained in the serum to attach to the glycans on the slide. A primary layer of

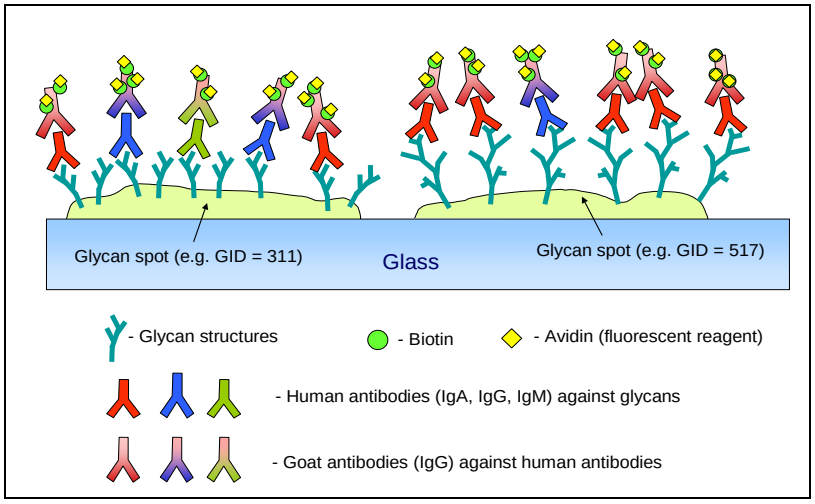
Glycan spot (e.g. GID = 311)
Glass
Glycan spot (e.g. GID = 517)
- Glycan structures
- Biotin
- Avidin (fluorescent reagent)
- Human antibodies (IgA, IgG, IgM) against glycans
- Goat antibodies (IgG) against human antibodies

Figure 2.2. Binding of the human antibodies and goat anti-human antibodies to the glycan structures on the PGA. Source: M. I. VUSKOVIC, H. XU, N. V. BOVIN, H. I. PASS, AND M. E. HUFLEJT, *Processing and analysis of printed glycan array data for early detection, diagnosis, and prognosis of cancers*. Unpublished report, 2011.

human IgG, IgM, and IgA immunoglobulins from the serum bind directly to the glycans on the slide. A secondary layer of biotinylated goat anti-human IgG, IgM, and IgA antibodies created by Pierce Biotechnology, Inc. attach to the human immunoglobulins. Avidin, a fluorescent reagent developed by Invitrogen/Molecular Probes, is bound to the goat anti-human antibodies.

Once the antibody binding is complete, the PGAs are scanned by a laser at a predetermined power and the signal intensities are read and measured using ImaGene software, developed by BioDiscovery, Inc. Figure 2.3 shows an image from the laser scanner showing one sub-array of a PGA [7].

The right side of the diagram shown in Figure 2.4 details the printing, developing, scanning, and quantification of the PGAs. The GlycoAnalyzer controls the Data Preprocessing and Data Analysis steps on the left side of the diagram. The rest of this thesis will discuss these steps and how they are integrated into the GlycoAnalyzer application.

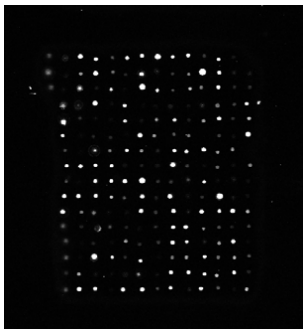

Figure 2.3. Image from a developed PGA sub-array. Source: M. I. VUSKOVIC, H. XU, N. V. BOVIN, H. I. PASS, AND M. E. HUFLEJT, *Processing and analysis of printed glycan array data for early detection, diagnosis, and prognosis of cancers*. Unpublished report, 2011.

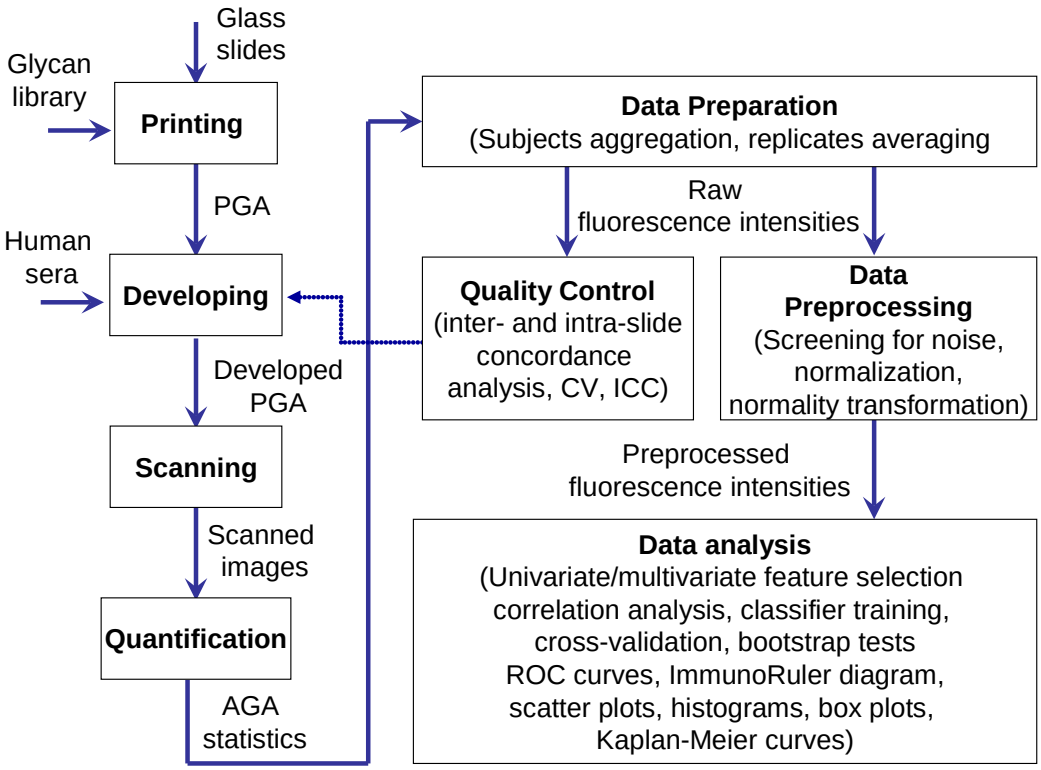

Figure 2.4. Steps in preparing and processing the PGA and the steps involved in the data analysis. Source: M. I. VUSKOVIC, H. XU, N. V. BOVIN, H. I. PASS, AND M. E. HUFLEJT, *Processing and analysis of printed glycan array data for early detection, diagnosis, and prognosis of cancers*. Unpublished report, 2011.

CHAPTER 3

DATA PROCESSING AND ANALYSIS USED IN GRAPHICAL DATA ANALYSIS

This section discusses the preprocessing, feature selection, projection, and plotting concepts that define the functionality of the GlycoAnalyzer application. Once the data is pulled from the PGA slides and loaded into a single, formatted MATLAB binary MAT-file, it can be loaded into the application and data processing can begin.

3.1 BACKGROUND

Each patient has separate PGA slides that are created over several batches and each slide is quantified individually. The first step in this process is the visual examination of each image that is created using the ImaGene software for noticeable imperfections and defects. Some of these defects may include, but are not limited to oddly-shaped spots and scratches or other noise that can be determined by visual inspection. If any defects are found in a particular image, the slide is discarded and the process of developing and reading a patient's slide is started again. If the slide is accepted, the data is loaded into a binary MAT-file. This file contains two separate matrices of information for the patient. One matrix is of the total fluorescence intensity at a concentration of 10μM and the second matrix is of the total fluorescence intensity of 50μM. Mean intensities could be used, but it has been found that total intensity does a better job of displaying the binding level of AGA. Using total intensities instead of mean intensities is also more valid because the distribution of glycans on each PGA is regular. To determine this, "salt images" of the glycan distributions are checked on each slide as soon as the creation of the slide is completed [7].

The second step in slide quality control has to do with the reproducibility of data between separate slides between batches for each patient and within sub-arrays on each slide. The former is for inter-slide quality control and the latter is for intra-slide quality control. Lin's concordance coefficient is used to determine the quality of the data from slide to slide [10]. The equation for this is:

$$CCC = \frac{\rho}{1+\frac{(s_1{}^2 - s_2{}^2)+(m_1-m_2)^2}{2s_1 s_2}} \tag{3.1}$$

This equation takes into account the Pearson correlation coefficient, ρ:

$$\rho = \frac{s_{12}}{s_1 s_2} \tag{3.2}$$

where the calculated means, variances, and covariances for each similar glycan over two slides are $m_k = \text{mean}_j(\tilde{x}_{kj})$, $s_k = \text{var}_j(\tilde{x}_{kj})$, $s_{12} = \text{cov}_j(\tilde{x}_{1j}, \tilde{x}_{2j})$ and the fluorescence intensities of the antibodies is $\tilde{x}_{kj}$ for sample index $k = 1,2$ where $Control\ set = 1$ and $Case\ set = 2$. Finally, j is the glycan index, where: $j = 1,2, \dots, d$ $(d = 211)$. Raw signals are expressed by using the tilde symbol (~) [7]. The Pearson coefficient relates each measurement to a best fit line and is used as a "measure of precision" [11]. The inter-slide quality control is only conducted on some of the slides due to the price if each individual slide. The requirements for slides being tested are that the serum for the patient must be processed on two separate days and that each slide must be from a different batch. For a slide to be accepted it must have a CCC > 0.85 and a CCC between 0.85 and 1.0 is considered normal [7].

Intra-slide quality control involves the reproducibility of data between different matrices on the same slide. The overall concordance coefficient is used for this test [12]. The equation for this test is:

$$OCCC = \frac{2\sum_{r=1}^{R-1}\sum_{k=r+1}^{R} s_{rk}}{(R-1)\sum_{r=1}^{R}\sigma_r{}^2 + \sum_{r=1}^{R-1}\sum_{k=r+1}^{R}(\mu_r - \mu_k)^2} \tag{3.3}$$

where R is the number of sub-arrays on a slide and μ_r, σ_r, and s_{rk} are the mean, standard deviation, and covariances of the replicates printed on a single PGA. Slides that have an OCCC < 0.9 are discarded and the same serum is used to develop slides until the calculated OCCC ≥ 0.9 [7].

Once all of the images from a study are accepted, the data from each patient's MAT-file file is combined into a single, large, binary MAT-file. This file contains two separate matrices of information. One matrix includes total fluorescence intensity data from all patients with a concentration of 10μM and the other matrix includes total fluorescence intensity data from all patients with a concentration of 50μM [7].

When the dataset structure, D, is loaded, it contains several matrices and arrays of information, including fields $D.X$, $D.GID$, $D.F$, $D.PID$, $D.P$, and $D.y$. The n by d matrix, $D.X$, contains the raw fluorescence intensity information read from the PGA slide. The 1 by d_{max} array, $D.GID$, contains the glycan numbers for the complete glycan library. The 1 by d array, $D.F$ contains the corresponding indices of array, $D.GID$, used in matrix $D.X$. The n_{max} by 1 array, $D.PID$, contains the patient identification strings for each patient with data in a particular study. The n by 1 array, $D.P$, contains the corresponding indices of array, $G.PID$, for patients listed in the matrix, $D.X$. The Finally, the n by 1 matrix, $D.y$, contains the class labels for each of the patients in a particular study. The number of matrices and arrays are doubled in the structure, D, because the dataset contains information for both sets of fluorescence intensities, 10μM and 50μM. Figure 3.1 shows a graphical representation of one of the two available sets of data.

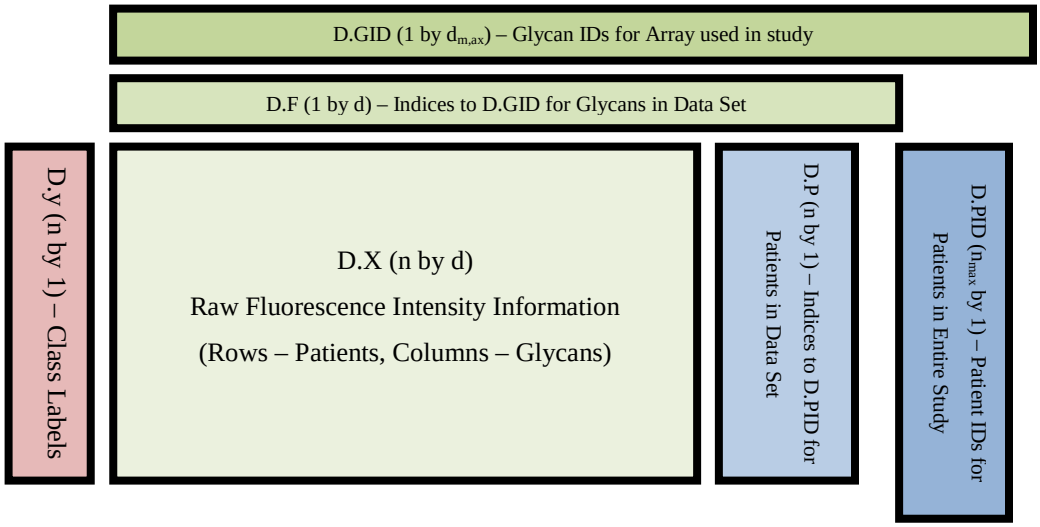
D.GID (1 by d_max) – Glycan IDs for Array used in study
D.F (1 by d) – Indices to D.GID for Glycans in Data Set
D.y (n by 1) – Class Labels
D.X (n by d)
Raw Fluorescence Intensity Information
(Rows – Patients, Columns – Glycans)
D.P (n by 1) – Indices to D.PID for Patients in Data Set
D.PID (n_max by 1) – Patient IDs for Patients in Entire Study

Figure 3.1. Graphical representation of the raw dataset packed in structure *D*.

3.2 DATA PREPROCESSING

Once the patient data has passed the visual, inter-slide, and intra-slide quality control phases, it can be loaded into the GlycoAnalyzer in a single binary MAT-file. This data still contains information that requires preprocessing to make it more convenient for patient analysis. The preprocessing phase consists of noise screening, normalization, and normality transformation to reduce the number of unreliable glycans.

Noise screening involves stripping the data of all glycans below or above certain threshold levels. One way this is done is to drop all glycans with low fluorescence intensities using the following equation:

$$\sum_{i=1}^{n} I\left(\tilde{x}_{ij} \leq s_{\propto}\right) \geq n - k \tag{3.4}$$

In this equation, j represents the glycan in question, $I(E)$ represents the indicator function over predicate E, i is the patient, n is the number of patients, k is the amount of aggressiveness used in screening, and $s_{\propto}$ is the noise threshold. The noise threshold for all replicates can be calculated using the equation:

$$\frac{1}{nd}\sum_{i=1}^{n}\sum_{j=1}^{d} I\left(\delta_{ij} \leq s_{\alpha}\right) \geq 1-\propto \tag{3.5}$$

Where δ is either the standard deviation of replicates or the median absolute deviation (MAD) for all replicates for patient i and glycan j and α a noise screening variable (e.g. $\alpha = 0.05$). MAD can be calculated using:

$$\delta_{ij} = \mathrm{MAD}_r\left(\tilde{x}_{ij}^{(r)}\right) = \left|\mathrm{median}_r\, \tilde{x}_{ij}^{(r)} - \mathrm{median}_r\, \tilde{x}_{ij}^{(r)}\right| \tag{3.6}$$

where r is the replicated sub-array in question [7].

The second way to screen glycans for noise is to drop glycans that have a high coefficient of variation (CV) using:

$$CV_{ij} = \begin{cases} 0 & if\ \delta_{ij} = 0 \\ 1 & if\ \tilde{x}_{ij} < \delta_{ij} \\ \delta_{ij}/\tilde{x}_{ij} & otherwise \end{cases} \tag{3.7}$$

Glycans with a high CV can be rejected using the equation:

$$\frac{1}{nd}\sum_{i=1}^{n}\sum_{j=1}^{d} I\left(CV_{ij} \geq CV_0\right) = \beta \tag{3.8}$$

where CV_0 is a percentage of the coefficient of variation and β screening parameter [7].

The final way of screening glycans for noise is to drop all glycans below the threshold of the interclass correlation coefficient (ICC). The equation that estimates ICC is:

$$ICC = \frac{BSV}{BSV+WSV} \tag{3.9}$$

where BSV stands for Between Subject Variability and WSV stands for Within Subject Variability. The equation for BSV is:

$$BSV = BSV_0 - \frac{WSV}{r} \tag{3.10}$$

The equation for WSV is:

$$WSV = \text{mean}_i(\text{var}_k(x_{ik})) \tag{3.11}$$

The equation for BSV_0 is:

$$BSV_0 = \text{var}_i(\text{mean}_k(x_{ik})) \tag{3.12}$$

In these equations the values for x_{ik} are intensities for $i = 1, 2, \ldots n$ patients and $k = 1, 2, \ldots r$ replicates for a single feature. All glycans with ICC below the threshold are dropped, while all glycans above the ICC threshold are kept for data analysis [7].

Once noise screening is complete, data normalization can be used to reduce the systematic per-slide bias in scale and location [7]. For this study, global inter-array, linear normalization is used:

$$x_{ij} = \frac{\tilde{x}_{ij} - l_i}{s_i} \tag{3.13}$$

where $\tilde{x}_{ij}$ is the raw fluorescence intensity and x_{ij} is the normalized fluorescence intensity for patient i and glycan j. The variable l_i is the location parameter and the variable s_i is the scaling parameter determined by:

$$l_i = \text{mean}_{j \in J}\left(\tilde{x}_{ij}\right), s_i = \text{std}_{j \in J}\left(\tilde{x}_{ij}\right) \tag{3.14}$$

or alternately by:

$$l_i = \text{median}_{j \in J}\left(\tilde{x}_{ij}\right), s_i = \text{MAD}_{j \in J}\left(\tilde{x}_{ij}\right) \tag{3.15}$$

In these equations, J is a set of column indices for glycans that are still left after the initial noise screening preprocessing phase. For the mesothelioma data set, most of the glycans are class independent. In fact, approximately 90 percent of the glycans on the mesothelioma PGAs are found to be class independent, making this procedure a good way to reduce linear bias in the remaining glycans with minimal damage to discriminatory information [7].

Finally, normality transformation is used to shorten the tails of the distribution for the remaining glycans. For this, the Box-Cox method was selected and has been extended to accept values that are negative [13]:

$$f(x, \lambda) = \begin{cases} \frac{\sin(x)\left((|x|)^{\lambda} - 1\right)}{\lambda}, \lambda \neq 0 \\ \sin(x)\log(|x| + 1), \quad \lambda = 0 \end{cases} \tag{3.16}$$

where λ is the power transform parameter. In studies with Mesothelioma patients, it was determined that $\lambda = 0.2$ gave best results. This value was determined after careful experimentation with actual and artificial data [7].

3.3. MEASURING THE GOODNESS OF DISCRIMINATION

The main goal of the functions used in the GlycoAnalyzer application is to provide ways of processing patient data pulled from PGAs. The idea behind the GlycoAnalyzer application is to provide an easy-to-use tool for non-programmers to be able to run the functions from an ordinary PC using a self-contained graphical user interface instead of running the functions from the MATLAB Command Window. The GlycoAnalyzer application provides a full set of data analysis algorithms which allow scientists and medical doctors to read in patient training data, process it, and, make predictions for additional unknown patient data.

Once the training data is loaded the noisy features have been removed, the classification algorithms in the Feature Selection and Projection Controls section of the application will allow researchers to specify classification algorithms that will identify the differences between the control and case sets. Once the identification of the selected features is complete, the selected feature set and classification algorithm should be able to make predictions and correctly classify unknown features that are included in test data gathered from completely different sources [7].

3.3.1 Student and Wilcoxon Statistic

Student's t-test and the Wilcoxon statistic are the first two feature selection methods used in the GlycoAnalyzer application. Both of them can be selected using the Feature Selection pull-down menus in the Feature Selection and Projection Controls section of the GUI.

Student's t-test is a common approach used to determine if the means of two independent, nearly normally distributed groups of patients, the control and the case groups, differ statistically. The t-test can be calculated with each of the sample group's means, standard deviations, and number of data points. In the GlycoAnalyzer application, the unpaired t-test is used, because there is not always the same number of points in each of the sample groups [14]. The t-test is a signal to noise ratio calculation and can be calculated as follows:

$$\frac{Signal}{Noise} = \frac{Difference\ between\ the\ means\ of\ two\ groups}{Variability\ of\ the\ two\ groups} = tvalue \tag{3.17}$$

or

$$tvalue = \frac{|\bar{X}_1 - \bar{X}_2|}{\sqrt{P*Q}} \tag{3.18}$$

In this equation $\bar{X}_1$ and $\bar{X}_2$ are the sample means of the selected control and case groups. P and Q can be calculated as follows:

$$P = \frac{(n_1+n_2)}{n_1 n_2} \tag{3.19}$$

and:

$$Q = \frac{[(n_1-1)s_1+(n_2-1)s_2]}{[n_1+n_2-2]} \tag{3.20}$$

In the equations for P and Q, n_1 and n_2 are the number of sample data points in the control and case groups respectively, and s_1 and s_2 are the standard deviations for each group. A higher t-value represents a larger difference between the two groups [15].

The Wilcoxon rank-sum test can be used as an alternative to the Student's t-test when the user cannot assume or determine if the samples are normally distributed. Like Student's t-test, the Wilcoxon rank-sum test is calculated by comparing different measurements between two groups of patients. Unlike the t-test where the mean and standard deviations of the two sample sets are used to compare the sets, the Wilcoxon rank-sum test combines the values in the two sets, assigns a rank to each observation based on where they fall in relation to one another, and then compares the ranks of the observations to determine a difference between the two sets [16].

If the control set, A, has a number of distinct observations, n_A, and the case set, B, has a distinct number of observations, n_B, and both of these groupings of observations are independent of each other, the Wilcoxon rank-sum test can be used to determine if the sets are the same or shifted from one another. The variable, H_0, is the null hypothesis that the distribution of scores for each set is identical:

$$H_0 : A = B \tag{3.21}$$

The variable, H_1, is the alternate hypothesis that the distribution of scores for each group is not identical. There are three ways to write this hypothesis:

$$H_1 : A < B \tag{3.22}$$

where the grouping of the control set, A, is shifted to the left of the case set, B.

$$H_1 : A > B \tag{3.23}$$

where the grouping of the control set, A, is shifted to the right of the case set, B.

$$H_1 : A \neq B \tag{3.24}$$

where it cannot be determined if the grouping of the control set, A, is shifted to the right or left of the case set, B. Figure 3.2 shows a graphical comparison of the difference between the hypothesis of H_0 and one of the possible hypotheses for H_1 [17].

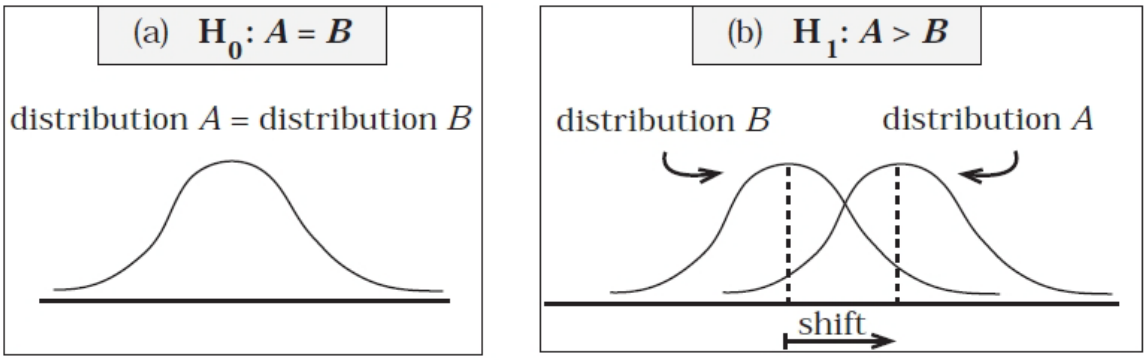

Figure 3.2. Graphical comparison between the hypotheses H0 and H1.

In order to conduct the Wilcoxon statistic for the control group, W_A, all of the numerical observations from each group are combined in order in a single group. Once ordered, each observation is given a ranking from 1 to $n_A + n_B$. The observation with the smallest value is given the lowest value and the largest observation is given the largest value [16]. Once the ranking occurs, the sum of ranks from the control group is calculated so that:

$$w_A = \ sum\ of\ the\ ranks\ from\ set\ A \tag{3.25}$$

The two groups are assumed to have a continuous distribution so that:

$$\mu_A = \frac{n_A(n_A+n_B+1)}{2} \tag{3.26}$$

and:

$$\sigma_A = \sqrt{\frac{n_A n_B(n_A+n_B+1)}{12}} \tag{3.27}$$

where μ_A is the mean of the control group and σ_A is the standard deviation of the control group [18]. The p-value is the test of the rank sum, w_A against one of the hypotheses listed above, where:

$$pvalue = pr(W_A \geq w_A) \approx pr(Z \geq z) \tag{3.28}$$

and:

$$z = \frac{w_A+\mu_A}{\sigma_A} \tag{3.29}$$

Here, z represents the distance between the sample mean to the population mean in units of standard error. From this equation, the p-value can be calculated to test a hypothesis. From that value, it can be determined if the control group is shifted to the right or left of the case group [16].

3.3.2 Support Vector Machines

The support vector machine (SVM) is a machine learning algorithm developed by Vapnik that can be used in classification of data into two distinct classes [19]. The algorithm itself takes a set of classified training data and for new inputs, assigns the input to one of two classes based on the model created by the training data. In this algorithm, the set of training data is $X = X_1, X_2, \dots, X_n$ and the set of training labels is $y = y_1, y_2, \dots, y_n$. In the classification set, $y_i = 1$ if X_i is a member of the set and $y_i = -1$ if X_i is not a member of the set [20].

The purpose of SVM is to determine a hyperplane that separates both classes into two distinct groups of data. The position of the hyperplane maximizes the margin, *m*, or distance between the calculated hyperplane and the closest point of data in either set to the hyperplane. This hyperplane orientation is defined by a vector, *w*, which is perpendicular to the hyperplane. Figure 3.3 shows a graphical representation of the SVM concept [20].

Once a hyperplane is defined that has a margin to both sets, a new unknown set, X, can be run through the same algorithm while each example in the unknown set can be assigned to the two classes based on their location with respect to the hyperplane [20].

3.3.3 Receiver Operating Characteristic (ROC) Curve

When the individual features of two classes of patients are examined, one with a particular disease and one without the disease, there will rarely be a sharp distinction between the two sets. This can be due to any number of reasons, including; biological variations, equipment calibration errors, measurement errors, and environmental variations. The Receiver Operating Characteristic (ROC) curve analysis is a classifier evaluation model that can be used to assist in distinguishing between two sets of data at different points [21].

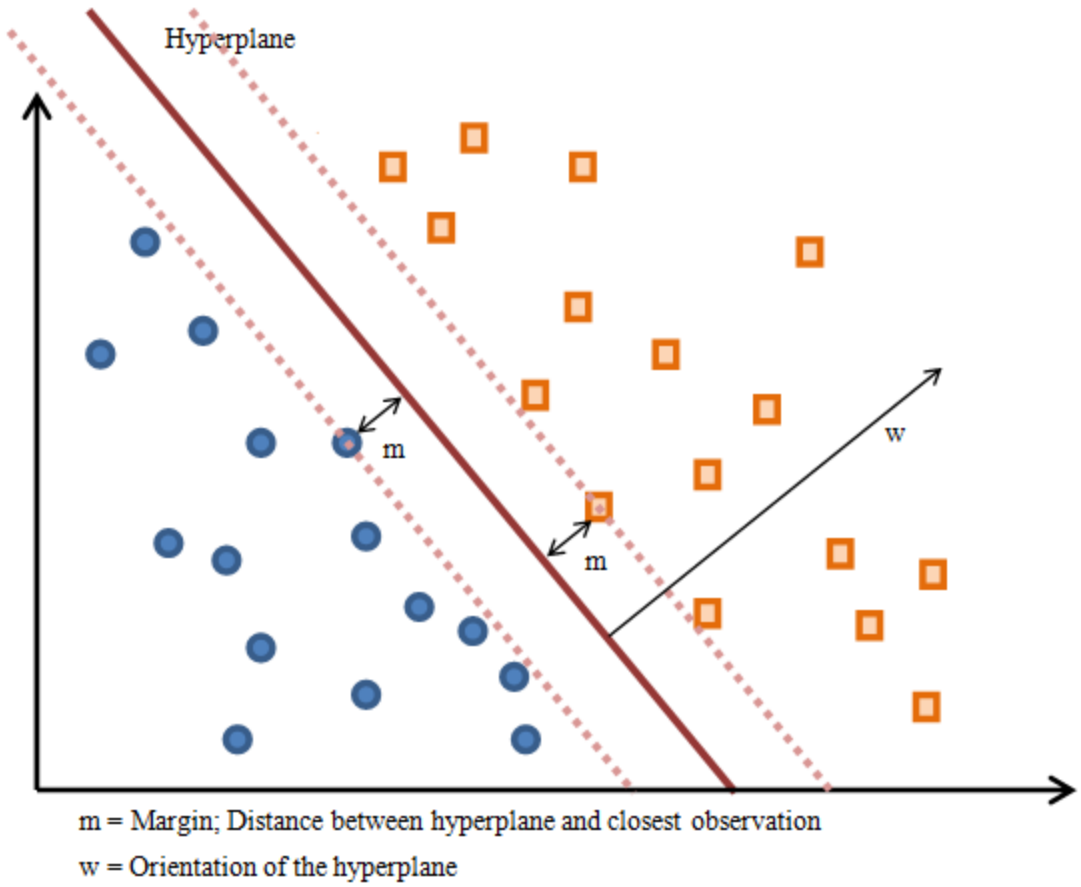

Figure 3.3. Graphical representation of the SVM concept.

Figure 3.4 displays a hypothetical plot of a specific feature. The measured feature, x, has a mean of μ_2 when the disease is present in a group of patients and a mean of μ_1 when the disease is absent. A threshold value, x^*, is used in deciding if a disease is present or not.

Four conditional probabilities can be determined from the plot shown above [21]:

1. $P(x < x^* \mid x \in \omega_1)$ or True Negative (TN) – Probability of correctly predicting that the patients did not have the disease.
2. $P(x < x^* \mid x \in \omega_2)$ or False Negative (FN) – Probability of incorrectly predicting that the patients did not have the disease.
3. $P(x > x^* \mid x \in \omega_2)$ or True Positive (TP) – Probability of correctly predicting that the patients had the disease.
4. $P(x > x^* \mid x \in \omega_1)$ or False Positive (FP) – Probability of incorrectly predicting that the patient had the disease.

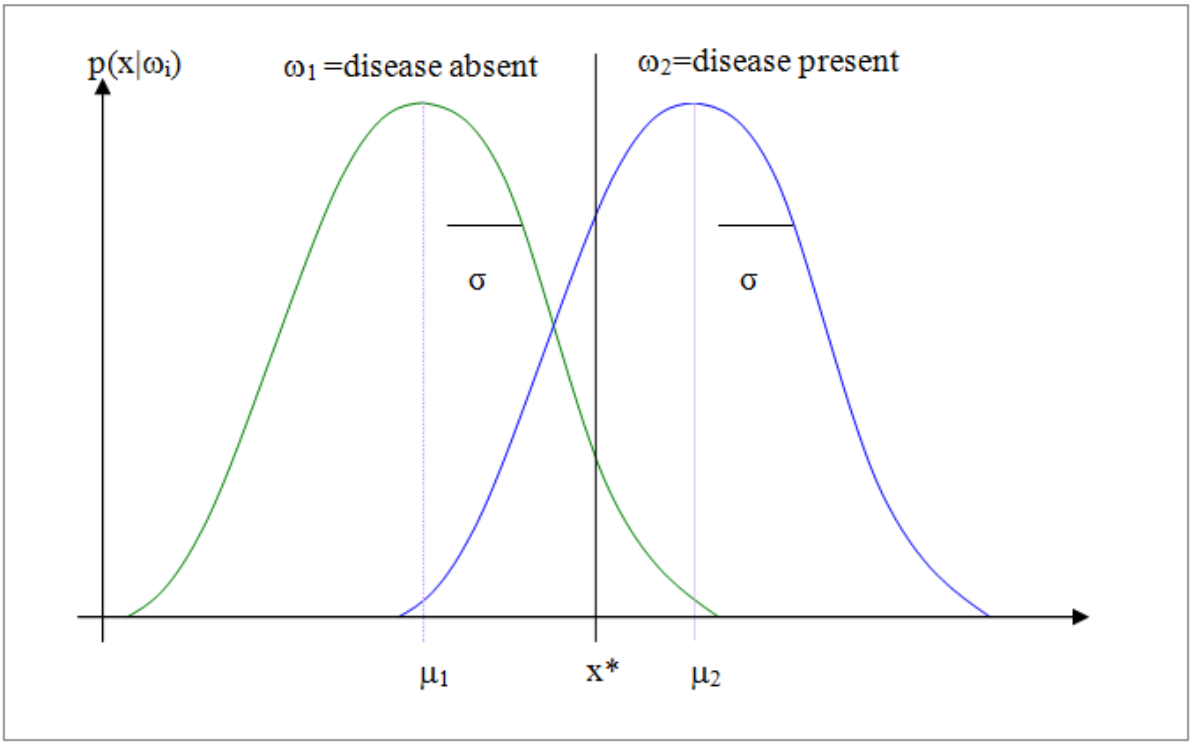

Figure 3.4. Hypothetical plot of a specific feature.

3.3.4 Specificity and Sensitivity

Table 3.1 is another way of displaying the information listed in the four conditions listed above. From this contingency table, several statistics can be calculated for each threshold value, *x**:

Table 3.1. ROC Contingency Table

Disease					
Test	Present	n	Absent	n	Total
Positive	TP	a	FP	c	a+c
Negative	FN	b	TN	d	b+d
Total		a+b		c+d	

The contingency table can be used to determine important quantities, such as sensitivity, specificity, the positive likelihood ratio, the negative likelihood ratio, the positive

predicted value, and the negative predicted value. The sensitivity is the probability that a disease will be correctly classified as occurring in a patient:

$$Sensitivity = \frac{a}{a+b} \tag{3.30}$$

The specificity is the probability that a disease will be correctly classified as not occurring in a patient:

$$Specificity = \frac{d}{c+d} \tag{3.31}$$

The positive likelihood ratio is a ratio of the true positive rate when the disease is present to the false positive rate when the disease is not present:

$$Positive\ likelihood\ ratio = \frac{Sesitivity}{(1-Specificity)} \tag{3.32}$$

The negative likelihood ratio is a ratio of the false negative rate when the disease is present to the true negative rate when the disease is not present:

$$Negative\ likelihood\ ratio = \frac{(1-Sensitivity)}{Specificity} \tag{3.33}$$

The positive predicted value is the ratio of the true positive rate to the total of the true positive rate and the false positive rate. Of all the true predictions, this value gives the percentage of the correct true predictions:

$$Positive\ predicted\ value = \frac{a}{(a+c)} \tag{3.34}$$

Finally, the negative predicted value is the ratio of the true negative rate to the total of the true negative rate and the false negative rate. Of all the false predictions, this value gives the percentage of the correct false predictions:

$$Negative\ predicted\ value = \frac{d}{(b+d)} \tag{3.35}$$

When a ROC curve is plotted, the plot consists of the sensitivity, or true positive rate (TP) verses 100-specificity, or the false positive rate (FP). The best possible case is that sensitivity and specificity are both plotted at 100%, meaning that patients having a particular disease were correctly classified 100% of the time as having the disease and that patients not having a particular disease will be correctly classified 100% as not having the disease. A successful test where all of the patients were correctly classified 100% of the time would have the curve touching the upper-left corner of the ROC curve. The closer the ROC curve reaches to the upper-left corner of the graph, the more accurate the analysis was. If the ROC

curve is close to a straight, diagonal line, (0, 0) – (1, 1), the data can be considered random [22].

3.3.5 Area Under the ROC Curve

The area under the ROC curve is a single-valued performance measure that can be used to determine the accuracy of certain features. The area under the ROC curve (AUC) can be computed as:

$$AUC = f_2(z, y) = \frac{S_2 - \frac{n_2(n_2+1)}{2}}{n_1 n_2} \tag{3.36}$$

In this equation, z is a linear combination of the projected intensities associated with selected features, y is a vector of the corresponding class labels, n_1 is the number of control samples, n_2 is the number of case samples, and S_2 is the sum of ranks of projected glycans for the case samples [23]. Each value of z_i represents the projected intensity of a single glycan. The equation:

$$z_i = x_i w \tag{3.37}$$

represents the combination where $w = (w_1, w_2, \ldots w_m)^T$ it the projection vector for the m selected glycans and $x_i = \left(x_{ij_1}, x_{ij_2}, \ldots x_{ij_m}\right)$ the row vector of the preprocessed fluorescence intensities for the m selected glycans [7]. AUC can be used to rank the performance of individual features because sample imbalances do not matter [24], the AUC values reflect the ranking of combined intensities rather than just binary decision [25], and the AUC value is not dependent on the choice of a decision threshold [26, 27]. Therefore, AUC is the preferred performance measure.

3.3.6 Adjusted ROC Curve

Ranking features once they have been selected can be done by adjusting the ROC curve using a compound feature selection method. Rather than just using the observed AUC as a basis for evaluating data and classifiers, the adjusted ROC curve uses a cross-validated evaluation that involves performing feature selection on groups of randomly selected subsamples from the control and case sets. The process for performing adjusted AUC is as follows:

1. Compute the observed ROC curve.

2. Perform a specified number of iterations for the following five steps:
3. Split the data into validation and training sets. This split must be done randomly.
4. Perform feature selection and projection based on the subsampled training set.
5. Create the ROC curve from the training set.
6. Create a ROC curve from the validation set.
7. Use the equation: $\Delta(fpr) = ROC_T(fpr) - ROC_V(fpr)$ to find the difference between the training and validation set curves.
8. Once the iterations are complete, find the average differences using the equation: $\Delta_{fpr}(fpr) = mean(\Delta)$.
9. Adjust the ROC curve using the equation: $ROC_A(fpr) = ROC_O(fpr) - \Delta_{avg}(fpr)$.

This algorithm reduces feature selection bias and generates an AUC value that is slightly higher than other methods, such as 10-fold cross-validation [7]. Figure 3.5 displays a ROC curve for the Mesothelioma assay. The solid blue line represents the ROC curve for the top 5 glycans, combined by multiple logistic regression. The dotted pink line represents the ROC curve for the single top feature. The solid red line represents the adjusted ROC curve for the top 5 features, determined by compound feature selection.

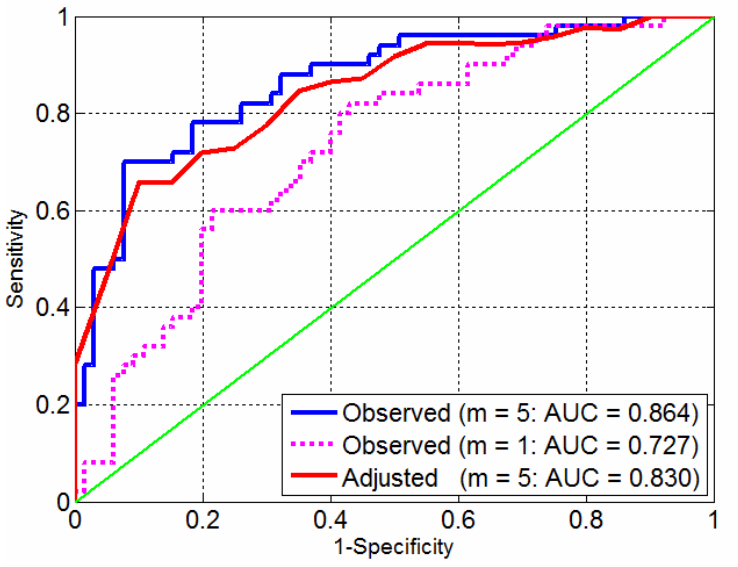

Figure 3.5. Sample ROC diagram for the mesothelioma assay displaying the adjusted ROC curve. Source: M. I. VUSKOVIC, H. XU, N. V. BOVIN, H. I. PASS, AND M. E. HUFLEJT, *Processing and analysis of printed glycan array data for early detection, diagnosis, and prognosis of cancers*. Unpublished report, 2011.

3.4 FEATURE SELECTION

Feature selection is the technique where a relevant subset of a larger group of features is selected and separated from other features that may not hold as much information. Once the numbers of features in the training set has been successfully paired down, the features selected during the feature selection process are used to, hopefully, successfully classify unknown patients.

Feature selection serves two purposes. First, if there is a large amount of initial training data, it helps reduce the amount of data into a more manageable set. Reducing the data reduces the time it takes to classify unknown patients. Second, the accuracy of classification often increases because while feature selection reduces the dataset, it also reduces the number of noisy features, increasing the accuracy of classifying new patients [28].

In the GlycoAnalyzer application, data from hundreds of patients is loaded in using a MATLAB M-file. Each one of these patients has 211 glycans associated with them [7]. Feature selection pairs down the large amount of glycans to a smaller set that can be used for classifying new patient data.

The feature selection algorithms generally fall into two classes: univariate feature selection methods and multivariate feature selection methods.

3.4.1 Univariate Methods

A univariate feature selection method is one that analyzes data using only a single feature at a time. During the feature selection process, each glycan is evaluated by some performance measure, such as the p-value or AUC-value. Once all of the glycans have been ranked, they are compared to each other to determine the top-ranked features. The data used in the GlycoAnalyzer application has an unknown distribution so a non-parametric univariate feature selection technique is desirable [7].

The GlycoAnalyzer application uses two univariate feature selection methods. These are the Student's t-test and the non-parametric Wilcoxon rank-sum test. Both of these methods can be selected in the GlycoAnalyzer using the Feature Selection pop-up menu in

the Feature Selection and Projection Controls section of the application (STUDENT and WMW). Once either of these univariate feature selection methods is selected in the GlycoAnalyzer, the application performs the feature selection.

3.4.1.1 STUDENT RANKING

If STUDENT is selected from the Feature Selection pop-up menu, the GlycoAnalyzer calls the functions *FS* and *T_sort_fast*. After a thorough argument check, the function *T_sort_fast* calls the MATLAB function, *ttest2*, from the Statistics toolbox.

The function *ttest2* performs a two-sample Student's t-test on the control and case vectors of data. For the GlycoAnalyzer, the t-test that is performed uses the value of alpha to indicate a rejection of the null hypothesis. In the case of the GlycoAnalyzer, this rejection is at a 5% significance level. The other two assumptions made by the t-test is that the means of the control and case sets are not equal and that the two sets do not have equal variances. Once the t-test is complete, the p-values, glycan indexes, and ranks are sorted and placed in a matrix for use by the GlycoAnalyzer [29].

3.4.1.2 WILCOXON RANKING

If WMW is selected from the Feature Selection pop-up menu, the GlycoAnalyzer application calls the functions *FS* and *W_sort*. First, the consistency of the arguments is checked. Checking is also done to ensure that there are only two classes. Once the data checking is complete, the function *W_sort* calls the MATLAB function, *ranksum*, from the Statistics toolbox.

The function *ranksum* performs a two-sided rank sum test on the control and case vectors of data and determines if the null hypothesis is a correct assumption for the data if the data is from two independent samples that have continuous distributions and equal means. The rejection of the null hypothesis is dependent on the variable alpha and is set at a 5% significance level. Once the Wilcoxon rank sum test is complete the AUC values are calculated for each of the features and the ranking is based either on the p-values. The ranks are sorted and placed in a matrix for use by the GlycoAnalyzer [30].

3.4.2 Multivariate Methods

While univariate feature selection involves the analysis of only one variable at a time, multivariate feature selection involves the statistical analysis of more than one variable at a time. This is a function of an n by d matrix of features, X, an n by 1 column vector of labels for those features, y, the number of features that are considered important, m, and the feature selection method used, φ. This function can be written as:

$$J_m = f(X, y, m, \varphi) \tag{3.38}$$

The multivariate feature selection techniques used in this application combine columns of matrix, X into a vector, z in the following way:

$$z = X(J_m)w_m \tag{3.39}$$

where $X(J_m)$ is a collection of combinations of features to be selected and w_m is a projection vector obtained by a projection method, such as Fisher linear discriminate, logistic regression, or a support vector machine, that is applied to $X(J_m)$ [7].

Multivariate feature selection methods often succeed when univariate feature selection methods fail. This is because single features may get poor rankings in univariate feature selection methods, but combined and evaluated with other combinations of features, they have a positive net effect on training. The dangers of multivariate feature selection include over-fitting and low cross validation with smaller sets of data [7].

The GlycoAnalyzer application uses seven multivariate feature selection methods. These feature selection methods are selected using the Feature Selection pop-up menu in the Feature Selection and Projection Controls section of the application and include:

1. Forward stepwise feature selection with logistic regression and resubstitution (FWD)
2. Feature selection based on recursive feature addition and projection based on the Fisher linear discriminant (RFA)
3. Feature selection based on recursive feature addition and projection based on the logistic regression (RFA_L)
4. Multivariate AUC-based recursive feature elimination with projection based on the fisher linear discriminate (RFE)
5. Feature selection based on recursive feature addition and projection based on the maximal projected margin (FFA)
6. Multivariate SVM-based recursive feature elimination with projection based on the recursive feature elimination algorithm proposed by Guyon and Elisseeff [31].

Additional methods will continue to be available in the application in the future as they are created. This paper will discuss specifically the RFE, GUYON, RFA, and RFA_L feature selection methods.

3.4.2.1 FISHER LINEAR DISCRIMINANT

The Fisher linear discriminate projection method is a way to classify multidimensional data. The first step is to project the data onto a single line in such a way that the distance between the means of the two sets is maximized, while the variance within each set is minimized. The equation for the projection vector determined by the Fisher criterion is defined as:

$$w = {S_p}^{-1}(m_1 - m_2) \tag{3.40}$$

where:

$$S_p = \frac{(1-m_1)S_1+(1-m_2)S_2}{m_1+m_2-2} \tag{3.41}$$

The w is the linear projection vector, m_k are class means, and S_1 and S_2 are covariance matrices for the control and case groups, and S_p is the pooled covariance matrix. Once the data is projected on the one-dimensional line, it can be divided into the two classes [32].

3.4.2.2 BACKWARD STEPWISE FEATURE SELECTION (RFE AND GUYON)

The GlycoAnalyzer application uses two separate recursive feature elimination algorithms. From the Feature Selection pop-up menu in the Feature Selection and Projection Controls section, these options are listed as RFE and GUYON in the menu. RFE is a multivariate AUC-based recursive feature elimination algorithm where projection is based on Fisher linear discriminant and GUYON is a multivariate SVM-based recursive feature elimination algorithm where projection is based on SVM. RFE is called from the function *FS* using the function *RFE_ROCMM_Fisher* and GUYON is called from the function *FS* using the function *RFE_GUYON*.

With backwards stepwise feature selection, iteration is used to remove features. Initially, the set of features contains every feature. Each time the algorithm goes through an iteration, the feature with the smallest ranking is removed until a determined amount of features remains.

3.4.2.3 FORWARD STEPWISE FEATURE SELECTION (RFA AND RFA_L)

The GlycoAnalyzer application uses two separate recursive feature addition algorithms. From the Feature Selection pop-up menu in the Feature Selection and Projection Controls section, these options are listed as RFA and RFA_L in the menu. RFA is a multivariate recursive feature addition algorithm where projection is based on the Fisher linear discriminate and RFA_L is a multivariate recursive feature addition algorithm with projection based on logistic regression. RFA and RFA_L are both called from the function *FS* using the function RFA. The only difference is that the projection method is different for each algorithm.

With forward stepwise feature selection, iteration is used to add features based on AUC value. Initially, the set of features is empty. Each time the algorithm goes through an iteration, the feature with the largest ranking is added until a determined amount of features is reached.

3.5 CLASSIFICATION

The main goal of the GlycoAnalyzer is to allow user to select different feature selection and projection algorithms, or classifiers, which will positively differentiate between the control and case sets of patients in a training dataset. Once this differentiation is determined, the goal is to look at a set of unlabeled data and using the set of selected top-ranked features and the classifier and be able to effectively classify the unlabeled data. Cross validation and bootstrapping techniques can be used to estimate how each classifier will perform. Once the training data is classified, the selected of the classifier must be validated using a second set of test data that was collected from a different source than the training data [7]. In the GlycoAnalyzer, once the classification of the training data is complete, sets of validation data can be loaded and processed to validate that the classifier and projection method is valid and effective.

3.6 DATA VISUALIZATION

The GlycoAnalyzer application is able to plot data for the user using four different types of plots. The type of desired plot is selected using the Plot Type pop-up menu in the Plotting Controls section. The current choices for plotting data include:

1. IR New – ImmunoRuler plot with integrated box plot
2. IR – Basic ImmunoRuler plot
3. PDF plot
4. ROC plot

Selecting any of the plot types automatically changes the available pop-up menus, radio buttons, and push buttons in the Plotting Controls section so that the visible controls are appropriate for the selected plot type. This, hopefully, reduces confusion for the user as certain user controls in the Plotting Controls section are only useful for certain types of plots. Once the preprocessing, feature selection, and projection of data is complete, clicking the Plot push button displays the plot of the data simultaneously in both the Main window and the Plot window. Future versions of the GlycoAnalyzer will include other types of plots, including scatterplots, box plots, and dot plots.

3.6.1 ImmunoRuler Plots

The ImmunoRuler plot, proposed by Vuskovic and colleagues [7, 33], is a convenient display of the results once the selection of optimal features is complete and the projection vector is calculated. Figure 3.6 [7] depicts a sample ImmunoRuler plot. The ImmunoRuler plot is a color coded bar graph that sorts patients based on a risk score. Figure 3.6 depicts a sample ImmunoRuler plot. Figure 3.6 depicts a sample ImmunoRuler plot. The left group contains subjects in the Control group and the right group contains subjects in the Case group.

The GlycoAnalyzer application allows for two types of ImmunoRuler plots; IR New and IR. The risk score for each patient in the training set is calculated and displayed using vertical colored bars. The risk score is calculated with the equation:

$$r_i = \frac{1}{1+\exp\left(-z_i - w_0\right)} \tag{3.42}$$

In this equation, r_i represents the risk score for each patient in the training set, z_i represents the projection and w_0 represents the classification decision point [7].

In the ImmunoRuler plot, the risk scores for each patient are separated for the control, case, and, in the case that validation data is loaded or the user selects any of the Test checkboxes, test sets. Each grouping is displayed with a different color where the control set is colored blue, the case set is colored red, and the test set is colored green. The order of risk

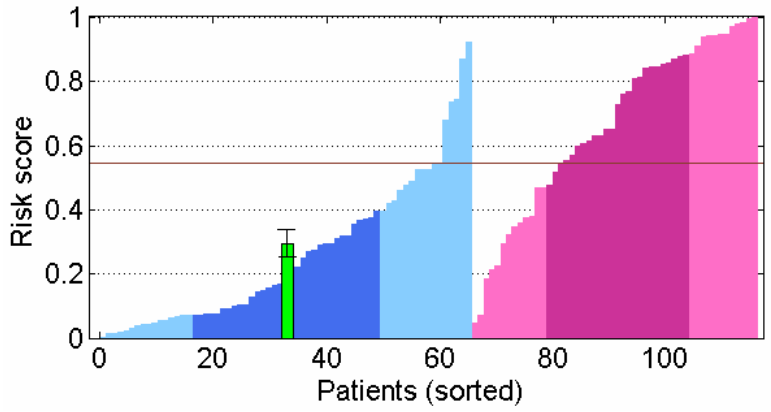

Figure 3.6. Sample ImmunoRuler plot. The bar graph with whiskers represents an unlabeled patient who is plotted against the control group. Source: M. I. VUSKOVIC, H. XU, N. V. BOVIN, H. I. PASS, AND M. E. HUFLEJT, *Processing and analysis of printed glycan array data for early detection, diagnosis, and prognosis of cancers*. Unpublished report, 2011.

sorting is controlled by the Sort pop-up menu in the Plotting Controls section of the application. The three sorting options are: ASCEND, DESCEND, and NONE. If the user selects NONE, the patient IDs are sorted from lowest to highest in each group.

Each ImmunoRuler plot also contains a threshold line which represents a decision point used for classification. In the GlycoAnalyzer, the threshold is changed using the Decision Point pop-up menu and the Cost editable textboxes. The Decision Point pop-up menu has four options: HMAX, MEAN, MEDIAN, and COST. When the cost option is chosen, the Cost editable textboxes appear allowing the user to specify integers between 1 and 100. When HMAX is selected, a threshold is selected that maximizes the training hit rate by calculating the number of correctly classified negative results, the number of incorrectly classified negative results, the number of incorrectly classified positive results, the number of correctly classified positive results, and the number of correctly classified patients at each possible threshold level for the two sets of projected training data and then selecting the threshold that maximizes the hit rate. Selecting MEAN finds the threshold by using the equation:

$$threshold = \frac{(mean(z_1)+mean(z_2))}{2} \tag{3.43}$$

where z_1 and z_2 are the projected data from the control and case classes of training data. When MEDIAN is selected, the threshold is calculated using the equation:

$$threshold = \frac{(median(z_1)+median(z_2))}{2} \tag{3.44}$$

Finally, selecting COST allows the user to enter numerical values for a ratio of the cost in miscalculating the controls versus the cost of miscalculating the cases in determining the optimum threshold. Cost of decision refers to the cost of a miscalculation used first by Niall Adams and David Hand in 1999 [34]. The equation:

$$L = \pi_1 f_1 C_1 + \pi_2 f_2 C_2 \tag{3.45}$$

is the calculated loss, where π_k is the probability of belonging to class k where $k = 1$ represents the control set and $k = 2$ represents the case group, f_k is the probability that class k will be misclassified, and C_k is the cost when class k is misclassified. This equation can be changed to:

$$nL = n_1(1 - s_p)C_1 + n_2(1 - s_n)C_2 \tag{3.46}$$

If we introduce the ratio of the cost of miscalculating the control class to the cost of miscalculating the case class:

$$\gamma = \frac{C_1}{C_2} \tag{3.47}$$

then minimizing the loss can be used to find a corrected decision point using the equation:

$$w_c = \text{argmax}_t[\gamma n_1 s_p(t) + n_2 s_1(t)] \tag{3.48}$$

where t is the value of the decision point. This maximization procedure is implemented by the ImmunoRuler function. The corrected decision point can be calculated using the equation:

$$r_c = \frac{1}{1+\exp(w_0 - w_c)} \tag{3.49}$$

The ImmunoRuler plot, IR New, can be used to classify a new patient who has not been classified. This is done by calculating the new patient's risk score using the selected features and the projection vector that is calculated during the training phase. The patient's risk score is plotted on the current ImmunoRuler plot with whiskers showing the standard deviation of the replicates [7]. The data from this patient is loaded in the Data Input Controls section using the Load Validation Data controls. This feature is not completed as of this writing, but will be finished in the next iteration of the GlycoAnalyzer application.

3.6.1.1 IMMUNORULER PLOT WITH QUARTILE REGIONS

The first of the two ImmunoRuler plots available in the GlycoAnalyzer is an ImmunoRuler plot with additional coloring that marks interquartile regions. The option for this plot is listed as IR New in the Plot Type pop-up menu. This version of ImmunoRuler only uses the risk scores from the control and case classes and does not enable the Test check boxes listed in the Feature Selection and Projection Controls section of the GlycoAnalyzer. If any of the checkboxes in the Test column is checked, that checkbox is ignored when the ImmunoRuler plot is created. Instead, this version of the ImmunoRuler plot can be used for classifying single unlabeled patients by loading in a MAT-file containing data for a single unknown subject in the Data Input Controls section of the application. This plot does not allow for validation to be loaded that contains data for more than one patient. A box plot for the unlabeled patient is placed in the correct spot of the controls class of the ImmunoRuler graph.

The controls and case class plots are separated into two colors, each representing a sample. In addition, the colors have two shades indicating the quartile ranges. The Control set is colored with a light blue/dark blue color combination and the Case set is colored with a light red/dark red color combination. If the Threshold radio button is selected, the threshold line can be varied each time the user clicks above or below the current threshold line. If the Patients radio button is selected, clicking on any of the bars produces a tool tip box that displays the patient's identification number and risk score. Once the plot is complete, the Training textboxes for the values Sp, Sn, PPV, NPV, ACC, and AUC are updated properly. Figure 3.7 displays a sample ImmunoRuler plot, IR New, without an unlabeled patient. Due to time constraints, the application has not been tested with data for a single unlabeled patient. Over the next few months, this functionality will be added to the application.

3.6.1.2 SIMPLE IMMUNORULER PLOT

The second of the two ImmunoRuler plots available in the GlycoAnalyzer application is a more general and simplified ImmunoRuler plot. The option for this plot is listed as IR in the Plot Type pop-up menu. This version of the ImmunoRuler plot allows for validation data

to be loaded in the Data Input Controls section that contains data for multiple patients and plots that data as a separate class from the control and case classes. It also allows for the

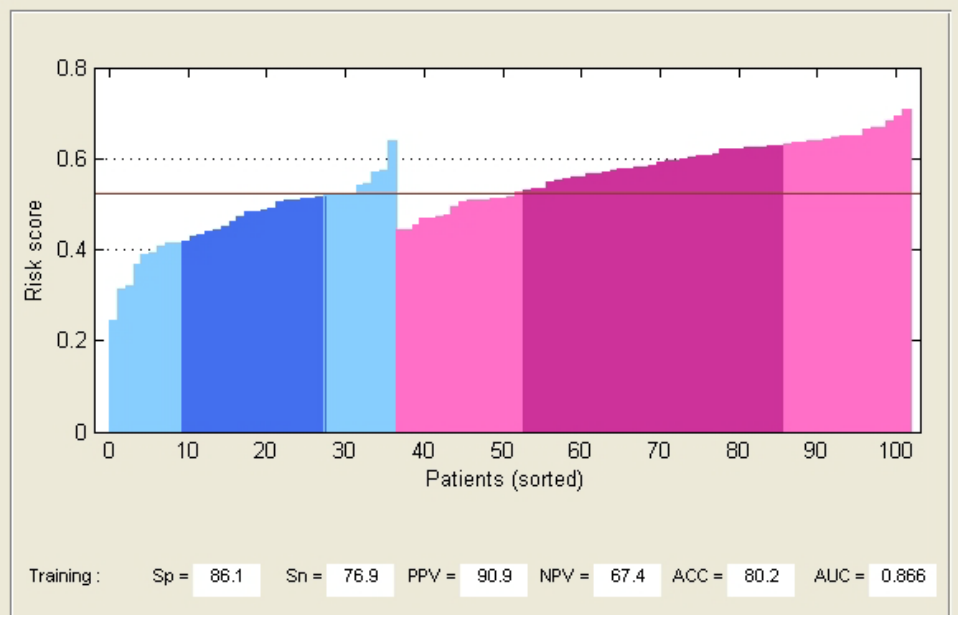

Figure 3.7. Sample ImmunoRuler plot, IR new.

enabling of the Test checkboxes listed in the Feature Selection and Projection Controls section of the application. If validation data is loaded, the Test column of checkboxes is made invisible so they cannot be selected by the user. If the validation data is deleted, the Test column becomes visible and selectable.

The controls and case class plots are separated into two colors, each representing quartile ranges. If validation data is loaded or any of the Test checkboxes are selected, a third color is displayed for this data representing a quartile range. If the Threshold radio button is selected, the threshold line can be varied each time the user clicks on the graph. If the Patients radio button is selected, clicking on any of the bars produces a tool tip box that displays the patient's identification number and risk score. Once the plot is complete, the Training textboxes for the values Sp, Sn, PPV, NPV, ACC, and AUC are updated properly. If validation data is plotted, the Validation textboxes for the values Sp, Sn, PPV, NPV, ACC, and AUC are updated properly. Figure 3.8 displays a sample ImmunoRuler plot, IR.

3.6.2 Probability Density Functions (PDF)

The GlycoAnalyzer application uses the MATLAB function, *ksdensity*, to plot the PDF function for each of the selected top features. In the GlycoAnalyzer function,

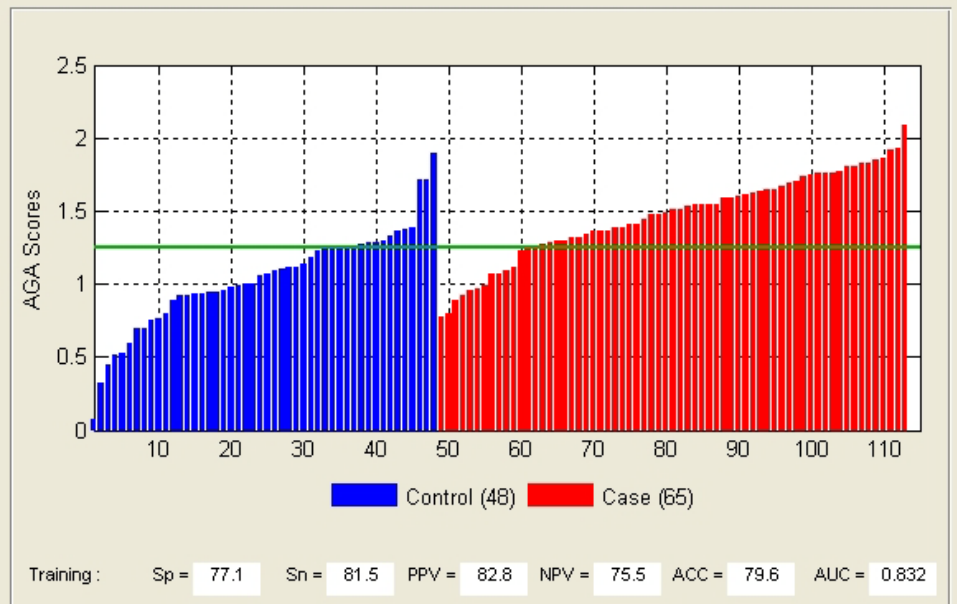

Figure 3.8. Sample ImmunoRuler plot, IR.

Two_PDF_GUI, the function, *ksdensity*, is used to calculate the kernel smoothing density estimate. In this function the projected vectors, z_1 and z_2, for the control and case classes are input as arguments. The outputs are p_1 and x_1 for the control class and p_2 and x_2 for the case class, where p_k is the vector of density values. Each value in the vector is evaluated at each of the points in the vector, x_k. The estimate is a normal kernel function and the width is calculated as a function of the number of points in the vectors, z_1 and z_2. The density function is evaluated over 100 points that are spaced equally over the entire range of z_1 and z_2 [35].

The PDF plot in the GlycoAnalyzer can be used in two different ways. By selecting INDIVIDUAL in the Plot Flag pop-up menu, each top-ranked feature is plotted on a separate graph in the Plotting Controls section (see Figure 3.9). The maximum number of individual features that can be plotted at a single time is six.

Each of these individual plots can be clicked to open a separate, larger plot in a figure outside of the application. By selecting COMBINED from the Plot Flag pop-up menu, the

information from each of the top selected glycans is combined and plotted on a single graph. Figure 3.10 displays a sample combined PDF plot.

In this sample, the control set is colored blue and the case set is colored red. This plot displays the top glycans and the p-value at the top of the graph.

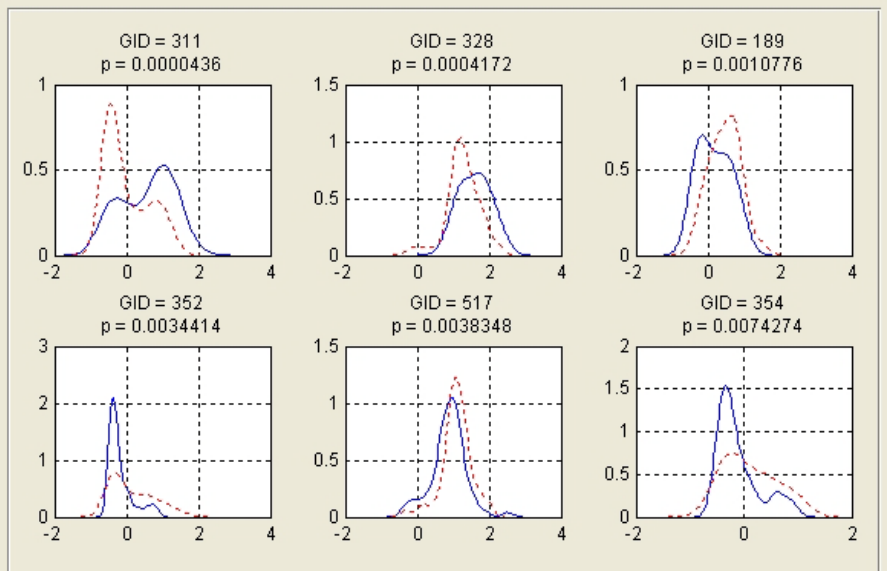

Figure 3.9. Sample individual PDF plots.

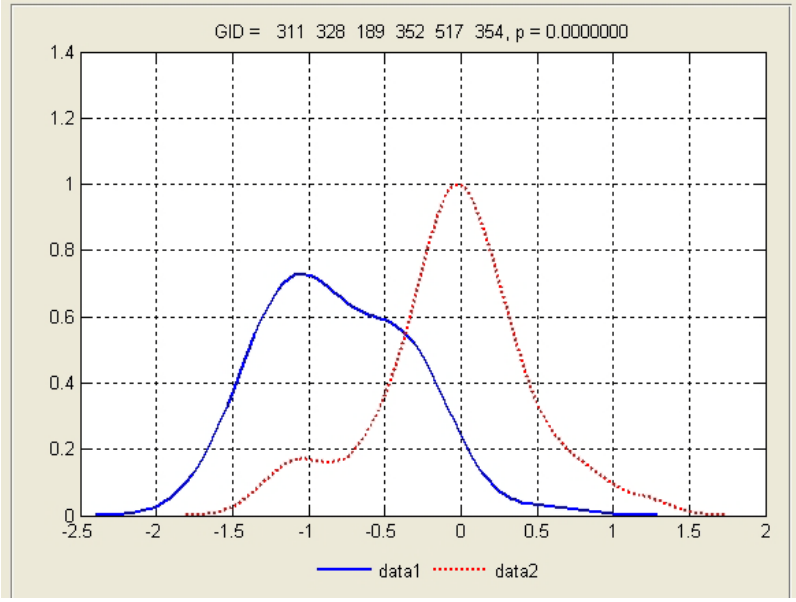

Figure 3.10. Sample combined PDF plot.

3.6.3 Receiver Operating Characteristic (ROC) Curves

As stated earlier, a ROC curve is a plot of sensitivity as a function of false predictive rate (100-specificity). In order to plot the information, the GlycoAnalyzer calculates sensitivity, specificity, and the area under the ROC curve using the function *ROC1* for individual glycans and the function *ROC_z* to calculate the same information for combined top features. Both functions determine the orientation of the sets of data with respect to each other and then calculate Sn and Sp by moving a threshold across various midpoints of adjacent observations and finding the number of true negative and true positive results.

The ROC plot in the GlycoAnalyzer can be used in two different ways. By selecting INDIVIDUAL in the Plot Flag pop-up menu, each top-ranked feature is plotted on a separate graph in the Plotting Controls section. The maximum number of individual features that can be plotted at a single time is six (see Figure 3.11). Each of these individual plots can be clicked to open a separate, larger plot in a figure outside of the application.

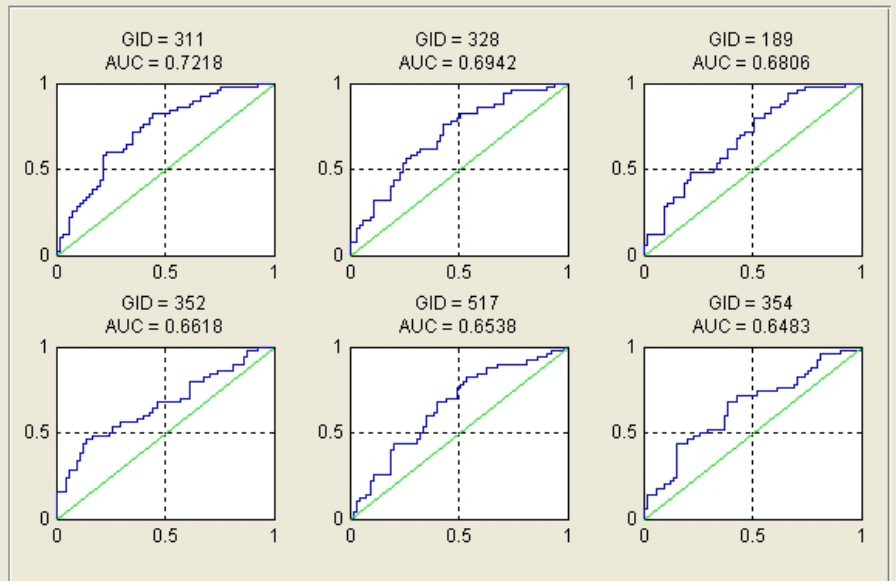

Figure 3.11. Sample individual ROC plot.

By selecting COMBINED from the Plot Flag pop-up menu, the information from each of the top selected glycans is combined and plotted on a single graph. Figure 3.12 displays a sample combined ROC plot. This plot displays the top glycans and the AUC-value at the top of the graph.

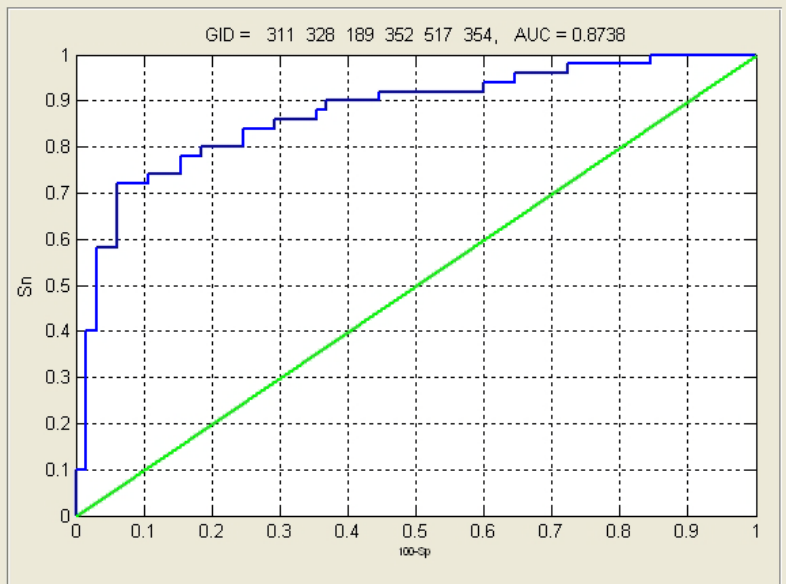

Figure 3.12. Sample combined POC plot.

CHAPTER 4

FUNCTIONALITY OF THE GLYCOANALYZER

This section specifies how the GlycoAnalyzer GUI is installed, launched, and used by potential users. The user interface is separated into four main windows: The Main window, the Preprocessing window, the Output window, and the Plot window. The Main window is used to input data files and labels, specify preprocessing, feature selection, and projection, and to provide a means for the visualization of data once the processing is complete. The Preprocessing window displays lists of glycans that have been removed once data preprocessing is complete. It also details brief reasons for why each glycan is removed. The Output window contains data related to the top ranked features once feature selection and projection have been completed. Finally, the Plot window is a mirror to the plotted data in the Main window axis and contains the same functionality, but it displays the data in larger axes.

4.1 INSTALLING THE GLYCOANALYZER APPLICATION

To install the GlycoAnalyzer application on a host PC, the user must first create a subdirectory called C:\GlycoAnalyzer. The GlycoAnalyzer application will be run directly from this location. The packaged application, *GlycoAnalyzer_pkg.exe*, must be copied and pasted into the GlycoAnalyzer subdirectory. Double-clicking on the packaged executable unpacks the components required by the executable and places them in the GlycoAnalyzer subdirectory.

The unpacked components include (1) *GlycoAnalyzer.exe*, (2) *ConfigFileHolder* folder, (3) *readme.txt* file, (4) *MCRInstaller.exe*. The *GlycAnalyzer.exe* file is the executable used to run the GlycoAnalyzer application. The *ConfigFileHolder* folder contains configuration files used by the application. The *readme.txt* file contains documentation on the deployment of the packaged application. The *MCRInstaller.exe* allows the application to be run outside of the MATLAB environment on any PC and is only required when the application is run for the first time on a new PC. Once *MCRInstaller.exe* is installed, it doesn't need to be installed again on the same PC. If desired, the user can create a shortcut

to the file *GlycoAnalyzer.exe* so that the application can be located easily. Figure 4.1 details the file structure of *GlycoAnalyzer_pkg.exe* as well as the file creation flow using *deploytool*.

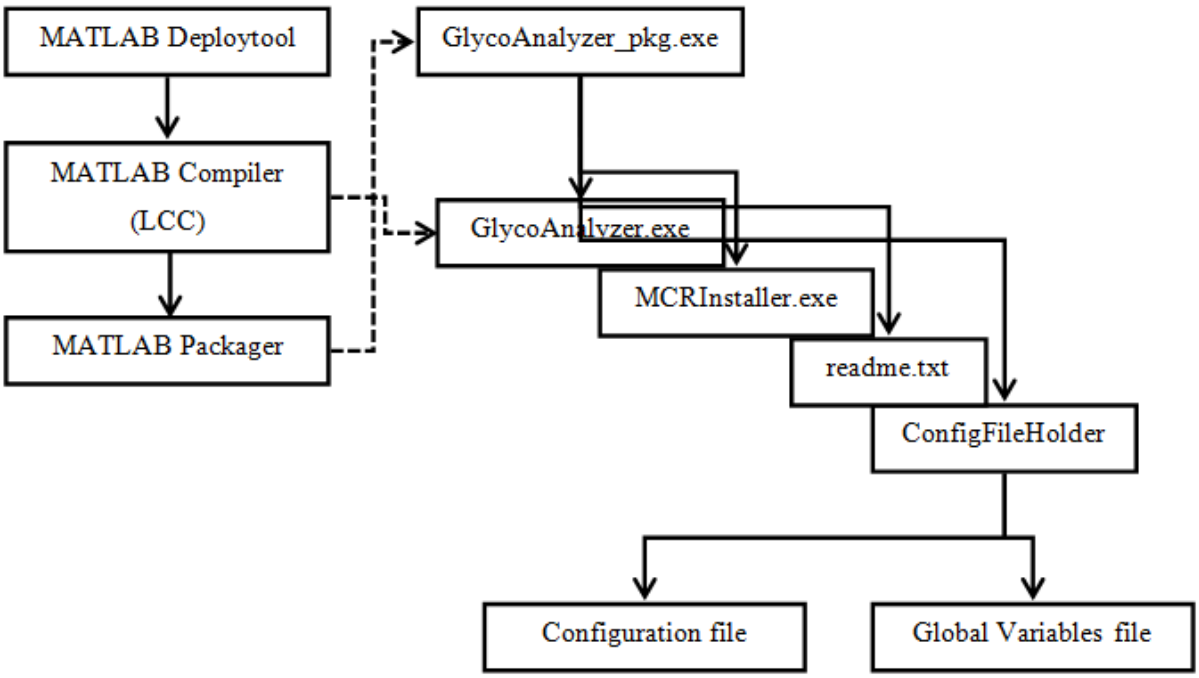

Figure 4.1. File structure of GlycAnalyzer_pkg.exe and file creation flow from deploytool.

4.2 Launching and Closing the GlycoAnalyzer Application

To launch the GlycoAnalyzer, the user double-clicks on the *GlycoAnalyzer.exe* file in the location: C:\GlycoAnalyzer. If this is the first time the application is run on a PC, the user is automatically prompted to install the MATLAB *MCRInstaller.exe* file. This file can be installed in the default location on the user's PC.

Initially, the GlycoAnalyzer Main window is displayed. If this is the first time the application is run, the application opens in the default initial state. In this state, the editable textboxes are populated with preset values, the non-editable textboxes are blank, and the pull-down menus are set to the first value in the list of possible values. If the application has been previously run on the same PC, the last user configuration is pre-loaded and all of the GUI components are set to the last known user values. Each time the GlycoAnalyzer is closed, the values from each of the GUI components are saved in a configuration file and reloaded the next time the application is launched by the user.

To exit the application, the user clicks the Close button in the lower-left corner of the application or by clicking the standard Windows Close button at the top-left corner of the application. In both cases, a Close dialog box appears stating, “Do you really want to close the application?” Figure 4.2 shows the Close dialog box. Clicking the Yes button saves all of the GUI component values and closes the application. Clicking the No button navigates the user back to the application.

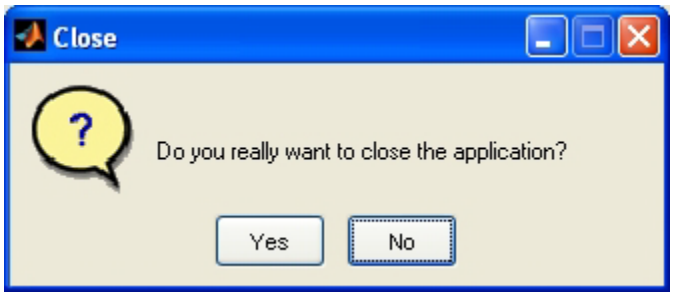

Figure 4.2. GlycoAnalyzer Close dialog box.

To open the Preprocessing window, the user clicks the View Data button in the Preprocessing section of the application. To close the Preprocessing window, the user clicks the Close button in the Preprocessing window. To open the Output window, the user clicks the View Data button in the Feature Selection and Projection section. To close the Output window, the user clicks the Close button in the Output window. To open the Plot window, the user clicks the Undock button in the Plotting section of the Main window. To close the Plot window, the user clicks the Dock button in the Plot window. When closed, all three windows are not actually closed, but merely invisible to the user. Opening and closing any auxiliary window involves a call to the window’s Visibility property. Once the section’s processing has been completed, the section’s window is populated with appropriate data. If no processing has been completed for the section, the associated window opens in a blank state.

4.3 Application Button Color Codes

The user is intuitively guided around the application by following the current colors of the buttons. If a button is highlighted in red, it is an indication of the next required step in the processing of data. Once the user completes the current step, the button associated with

the next step is highlighted in red. In Figure 4.3, the user initially sees the red highlighted Browse button next to the Load Training Data control.

Figure 4.3. Red Browse button before the training data is loaded.

Once the training data is successfully loaded, Figure 4.4 shows that the Browse button next to the Load Data Labels control was highlighted in red. If the user loads the validation data, the Browse button next to the Load Data Labels control would still be highlighted red because loading the validation data is not a required step for the GlycoAnalyzer application.

Figure 4.4. Red Browse button after the training data is loaded.

If the application is launched for the first time, the order of operation is as follows (1) Load the training data file, (2) Load the data labels file, (3) Complete the preprocessing of data, (4) Complete the feature selection and projection of data, (4) Plot the data. Each time the application is closed, the current configuration is saved. The next time the GlycoAnalyzer is launched, the previously saved configuration is loaded and the appropriate button is highlighted red indicating the starting step for the user. If the application is launched and only the training data file was loaded in the previous session, the Browse button next to the Data Labels control will be highlighted red indicating that loading the data labels file is the next step. If the training data and data labels files were loaded in the

previous session, the Run button in the Preprocessing section is highlighted red indicating that all of the appropriate patient data and data labels have been loaded from the previous session. Even if Feature Selection and Projection or Plotting was completed in the previous session, the Run button in the Preprocessing will be highlighted in red, indicating that preprocessing must be rerun each time the GlycoAnalyzer application is launched with the training data and data labels loaded during the previous session. This is to ensure each step is completed by the user when the application is launched.

After a configuration is run, if the training data file is changed or deleted, the Browse button next to the Load Data Labels file section is highlighted in red, forcing the user to load new data labels. After a configuration is run, if the data labels file is changed or deleted, the Run button in the Preprocessing section is highlighted in red forcing the user to run preprocessing using the new data labels with the current training data. If any component is changed in the Preprocessing or Feature Selection/Projection sections, the Run button in the same section is highlighted in red, forcing the user to rerun the processing in that section. If any component in the Plotting section is changed, the Plot button is highlighted in red forcing the user to re-plot the data.

4.4 Incorrect User Operations and Errors

Orange notifications are displayed if the user ignores the current highlighted button and proceeds to try a step that is out of sequence, enters a value in an editable textbox that is not an acceptable value, or does not check or incorrectly checks checkboxes in the Feature Selection and Projection section. When an orange notification is thrown, the area around the missing or incorrect information is highlighted in orange and a message with text describing the solution to the problem is displayed in the Status/Error textbox. Highlighting the area in orange directs the user to the specific area where the problem is occurring. The error displayed in the Status/Error textbox details the issue in writing for the user.

In Figure 4.5, the user attempted to load the data labels file before loading the training data file. The textbox to the right of the Load Training Data section is highlighted in orange directing the user to the problem area and a message directing the user to load the training data file is displayed in the Status/Error textbox.

Figure 4.5. Orange user error notification after an incorrect sequence of events.

In Figure 4.6, the user entered an incorrect value for the variable lambda in the Preprocessing Controls section. To direct the user to the incorrect value, the lambda textbox is highlighted in orange and a message detailing the acceptable values for the variable is displayed in the Status/Error textbox.

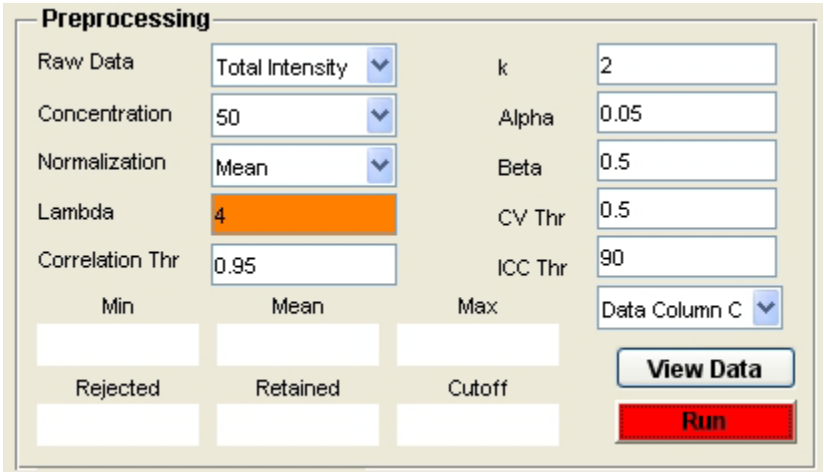

Figure 4.6. Orange user error notification after an incorrect value is entered in an editable textbox.

Run-time errors that occur in the programming of the GlycoAnalyzer application and are not caught by the application error handling are handled directly in the Status/Error section of the application. When an error is thrown because of a programming error a system error is thrown, the error text is displayed in the Status/Error textbox, and an orange "?" button becomes visible (see Figure 4.7).

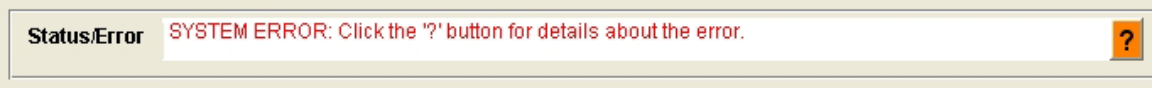

Figure 4.7. Orange "?" Button after a programming error has occurred.

When the user clicks the "?" button, a Generate Error dialog box appears which lists the filename and line number of where the error occurred in the application (see Figure 4.8).

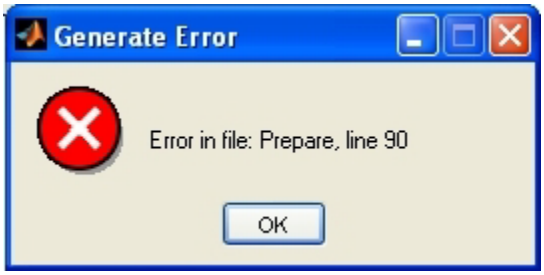

Figure 4.8. Generate Error dialog box.

This dialog box helps programmers who maintain/support the system determine exactly where errors are occurring in the code of the application. The GlycoAnalyzer uses hundreds of files to calculate patient data and finding the source of an error after the application has been released to users would be very difficult without this feature.

4.5 Main Window, Data Input Controls Section

The Data Input Controls section is where training data, validation data, and data label files are loaded and deleted. Application configurations can also be loaded and saved, making it possible for the user to call up previously saved configurations for different tests (see Figure 4.9).

Figure 4.9. GlycoAnalyzer Data Input Controls section.

To load a training or validation data file, the user clicks the Browse button to the right of the corresponding section. The standard Windows Open File dialog box appears allowing the user to browse for the desired binary MAT-file containing patient data. Once the file is located, it is properly loaded when the user clicks the Open button in the dialog box. To load a data labels file, the user clicks the Browse button to the right of the Load Data Labels

section. The standard Windows Open File dialog box appears allowing the user to browse for the desired XLS-file containing data labels. Once the file is located, it is properly loaded when the user clicks the Open button in the dialog box.

To delete the training data, validation data, or data labels file, the user clicks the Delete button to the right of the corresponding section. A question dialog box appears allowing the user to verify if he really wants to delete the file (see Figure 4.10).

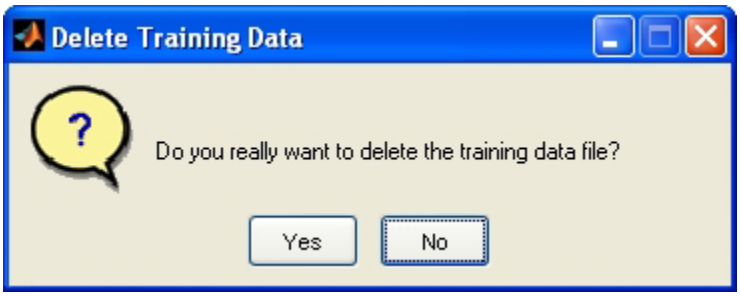

Figure 4.10. GlycoAnalyzer Delete File dialog box.

If the training data file is deleted, the data labels file is automatically deleted as well. This makes it easier for the user to load new training data that requires different labels. It also makes the user check the data labels each time a new training data file is loaded. Once the training data file is deleted, the Browse button next to the Load Training Data section is highlighted in red. If the data labels file is deleted, the Browse button next to the Load Data Labels section is highlighted in red. There is no change to the color of any of the Browse buttons when validation data is deleted.

To load a previously saved configuration file containing specific settings for each application component, the user clicks the Browse button to the left of the Load Config File section. The standard Windows Open File dialog box appears allowing the user to browse for the desired binary MAT-file containing GlycoAnalyzer configuration data. Once the file is located, it is properly loaded when the user clicks the Open button in the dialog box.

The configuration file contains saved values for every GUI component in the GlycoAnalyzer application. When the data in the configuration file is loaded, each component is updated with the value saved in the configuration file. If the configuration file was saved without training data or data labels, the Browse button to the right of the Load Training Data section would be highlighted in red. If only the training data was saved to the

configuration file, the Browse button to the right of the Load Data Labels section would be highlighted in red. If both the training data and the data labels were saved to the configuration file, the Run button in the Preprocessing section would be highlighted in red.

To save a snapshot of the current configuration of the GlycoAnalyzer at any point, the user clicks the Save Config button to the right of the Load Config File section. The standard Windows Open File dialog box appears allowing the user to name and save the file to any location. The configuration file is saved as a binary MAT-file in the user selected location. This file can be successfully loaded at any point once the GlycoAnalyzer application is running.

4.6 Main Window, Preprocessing Controls Section

The Preprocessing Controls section of the GlycoAnalyzer is where the initial screening of data occurs. It allows the user to filter out noisy data using noise screening, normalization, and normality transformation. The Preprocessing section contains editable textboxes and pop-up menus that allow the user to change the variables used during the preprocessing phase. Figure 4.11 shows the Preprocessing Controls section when the GlycoAnalyzer is opened for the first time or after the application has been reset. In this figure, each of the values for the editable textboxes and pop-up menus are set to initial default values.

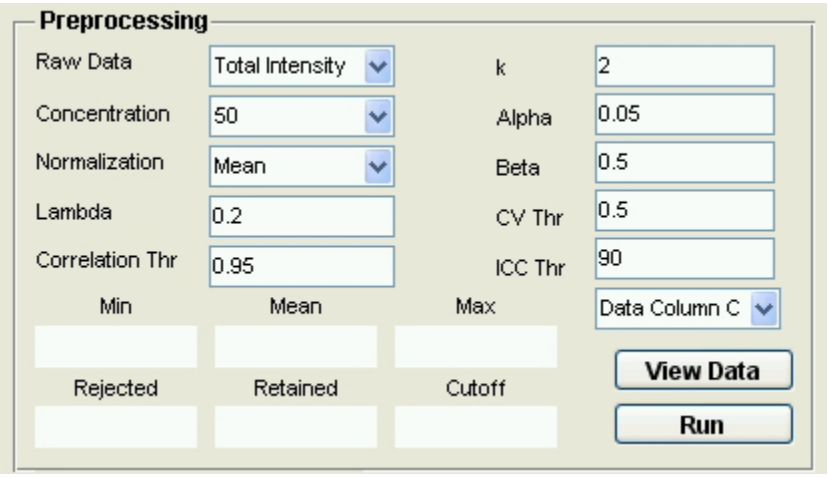

Figure 4.11. Preprocessing Controls Section with initial values.

The user may change any of the pop-up menus or editable textboxes prior to preprocessing. To begin the preprocessing of data, the user clicks the Run button. If any of the values in any of the preprocessing textboxes are outside of the designated limits, the textbox with the incorrect value is highlighted in orange to highlight the error and the Status and Error textbox displays an error message that details the proper limits for the user. The function of each Preprocessing Controls component and the correct values for each editable textbox are detailed in Appendix A.

Once the preprocessing stage is complete, the Min, Mean, Max, Rejected, and Retained non-editable textboxes are populated with the correct values and the Run button in the Feature Selection and Projection Controls section is highlighted in red. Currently, the Cutoff non-editable textbox is populated with the value, TBD, but will be correctly populated in a future version of the application (see Figure 4.12). If, at any time, after the preprocessing has been completed, the user changes any of the preprocessing values, the preprocessing Run button will be highlighted in red, signaling that the preprocessing phase must be run again.

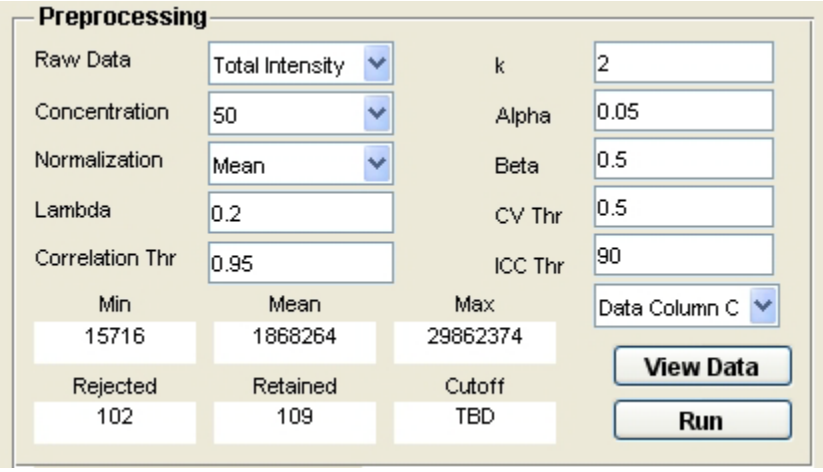

Figure 4.12. Preprocessing Controls after preprocessing is complete.

Once preprocessing is completed, the list of rejected glycans is displayed in the Preprocessing window of the application. To open the Preprocessing window, the user clicks the View Data button in the Preprocessing Controls section.

4.7 Main Window, Feature Selection and Projection Controls Section

The Feature Selection and Projection Controls section of the GlycoAnalyzer is where data analysis occurs on the glycans that remain after the preprocessing has occurred. The Feature Selection and Projection Controls section contains editable textboxes, pop-up menus, and checkboxes that allow the user to change the variables used during the feature selection and projection phases. Figure 4.13 shows the Feature Selection and Projection Controls section after the application is first opened. In this figure, the preprocessing values are set to the initial values and all Control, Case, and Test checkboxes are invisible.

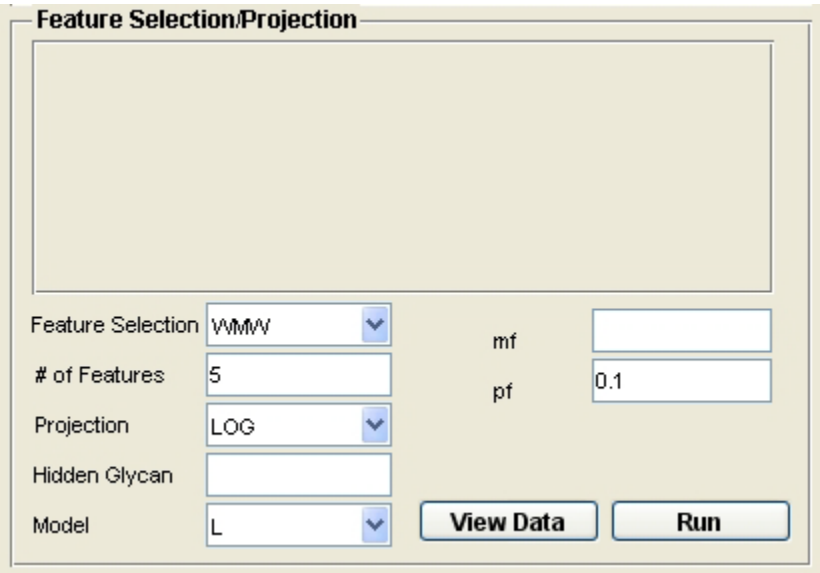

Figure 4.13. Feature Selection and Projection Controls before preprocessing.

Once preprocessing is complete, the labels from the data labels file populate the spaces next to each visible checkbox. The GlycoAnalyzer application can handle up to ten distinct data labels. If the data labels file contains four distinct labels, four checkboxes will be visible and selectable once preprocessing is complete (see Figure 4.14). Each data labels file contains three sets of data labels, the main assay and two subtypes of assays. The Column Select checkbox in the Preprocessing Controls section determines which set of labels populate the checkbox textboxes.

If validation data is loaded in the Data Input Controls section, the Test column of checkboxes will not be visible once preprocessing is complete. If a validation dataset is not

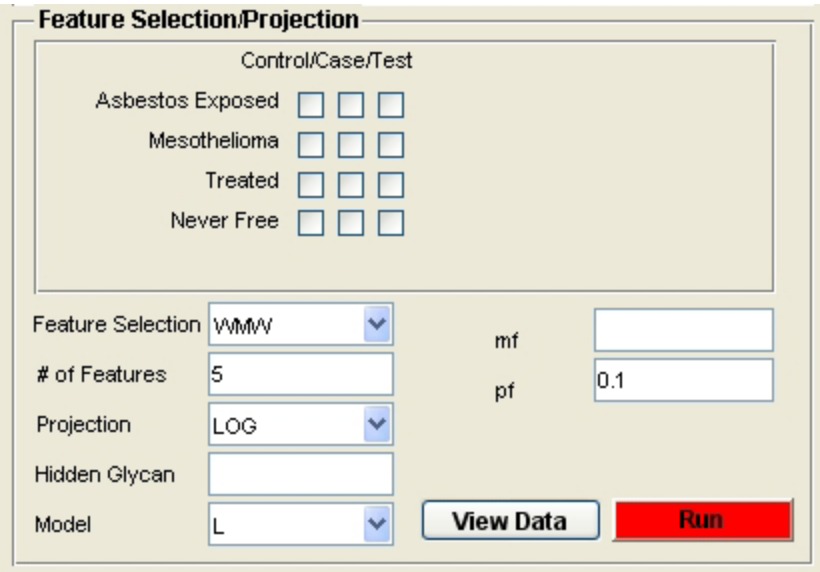

Figure 4.14. Feature Selection and Projection Controls after preprocessing.

loaded, the Test column checkboxes will be visible and selectable by the user. This is to prevent mixing actual patient validation data loaded from a validation data MAT-file and test data which is derived from the validation dataset MAT-file.

Each time the Feature Selection and Projection section of the application is run, at least one checkbox in the Control class column and one checkbox in the Case class column must be checked. Checking a checkbox in a particular column selects group of patients with a particular type of cancer. The control column selects the cancer classes for the control group of patients and the case column selects the cancer classes for the case group of patients. The same class cannot be checked in both the Control and Case columns, but multiple classes can be checked in both columns. If the Test column is visible, any checkbox can be checked even if the same class is checked in either the control or case column.

The *mf* and *pf* values are prefiltering values used during the feature selection process. Either value can be translated into the criteria for prefiltering, *mp*, as both textboxes are linked to each other. The variable, mp, represents the number of prefiltered candidate features which are used in the feature selection algorithm. The variable, *mf*, represents the number of Wilcoxon-ranked features that will be used in the feature selection process. The variable, *pf*, represents the number of Wilcoxon-ranked features for which the p-value of those features is greater to or equal to the value entered for *pf*. The user has the option to

enter a value for either *mf*, *pf* or both. The user can also leave both values blank. The values for mf and pf are translated into *mp* in the following way:

1. If the user enters a value for *mf*, but not *pf*: $mp = mf$
2. If the user enters a value for *pf*, but not *mf*: $mp = pf$
3. If the user enters values for both *mf* and *pf*: $mp = mf$
4. If the user does not enter values for *mf* and *pf*: $mp = 0$

If *mp* is equal to zero, no prefiltering is completed and all of the features that survived preprocessing are evaluated.

The Hidden Glycan textbox is for the user to enter a glycan number that will be evaluated regardless of prefiltering. Even if the feature is not one of the top features that remain after prefiltering, the hidden glycan is automatically included in the group of top features. This glycan is displayed in the list of top ranked features when the user opens the Output window of the GlycoAnalyzer application.

The function of the remainder of the Feature Selection and Projection Controls components and the correct values for each editable textbox are detailed in Appendix A. If, at any time, after the preprocessing has been completed, the user changes any of the values in the Feature Selection and Projection Controls section, the Run button for the section will be highlighted in red, signaling that the feature selection and projection phase must be run again.

Once feature selection and projection of data is completed, the list of top ranked features and information about those features is displayed in the Output window of the application upon the user's request. To open the Output window, the user clicks the View Data button in the Feature Selection and Projection Controls section.

4.8 MAIN WINDOW, PLOTTING CONTROLS SECTION

The Plotting Controls section of the GlycoAnalyzer allows the user to plot the results after preprocessing, feature selection, and projection of data is complete. The Plotting Controls section contains editable textboxes, pop-up menus, and radio buttons that allow the user to change the variables used during the plotting phase. It also allows the user to print the plot once it is complete.

The Plotting Controls section is actually broken up into two separate sections. The first section allows the user to select the type of plot, print the results, and open the Plot

window (see Figure 4.15) and the second section allows the user to change variables that modify the way the plot is displayed and displays the main axis of the plot (see Figure 4.16). Both sections are considered part of the Plotting Controls section. Initially, the main axis is blank. After plotting is complete, the user will see the selected type of plot displayed in the main axis.

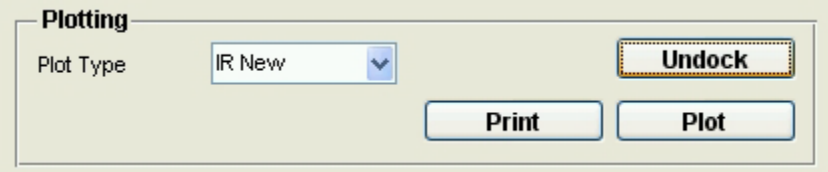

Figure 4.15. Plotting Controls allowing the user to select the plot type.

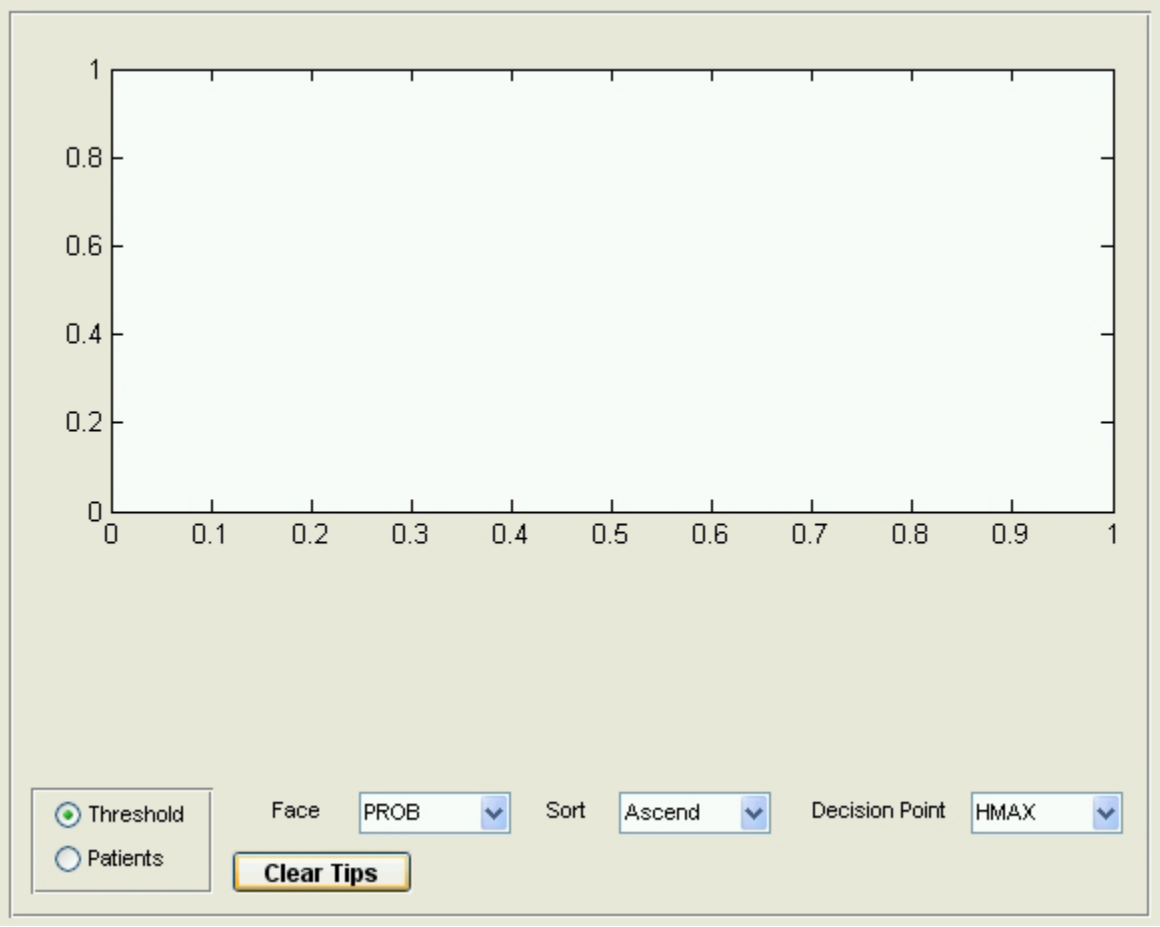

Figure 4.16. Plotting Controls for modifying and displaying the plot.

The four types of plots that are available to the user are two ImmunoRuler plots, a PDF plot, and a ROC plot. The details behind these plots are discussed in section 3.6. The two ImmunoRuler plots are interactive and let the user change the threshold line by clicking on the plot to change the height of the threshold line or display the patient identification number and risk score as a tooltip by clicking on an individual patient. The function of the remainder of the Plotting Controls components for the ImmunoRuler plots are detailed in Appendix A. The PDF and ROC plots allow the user to plot the top six features on up to six individual plots or on a single combined plot. The Plot Flag pop-up menu allows the user to change the plot from individual plots to a combined plot. Once the plot is complete, the selected plot is displayed in the main axis (see Figure 4.17).

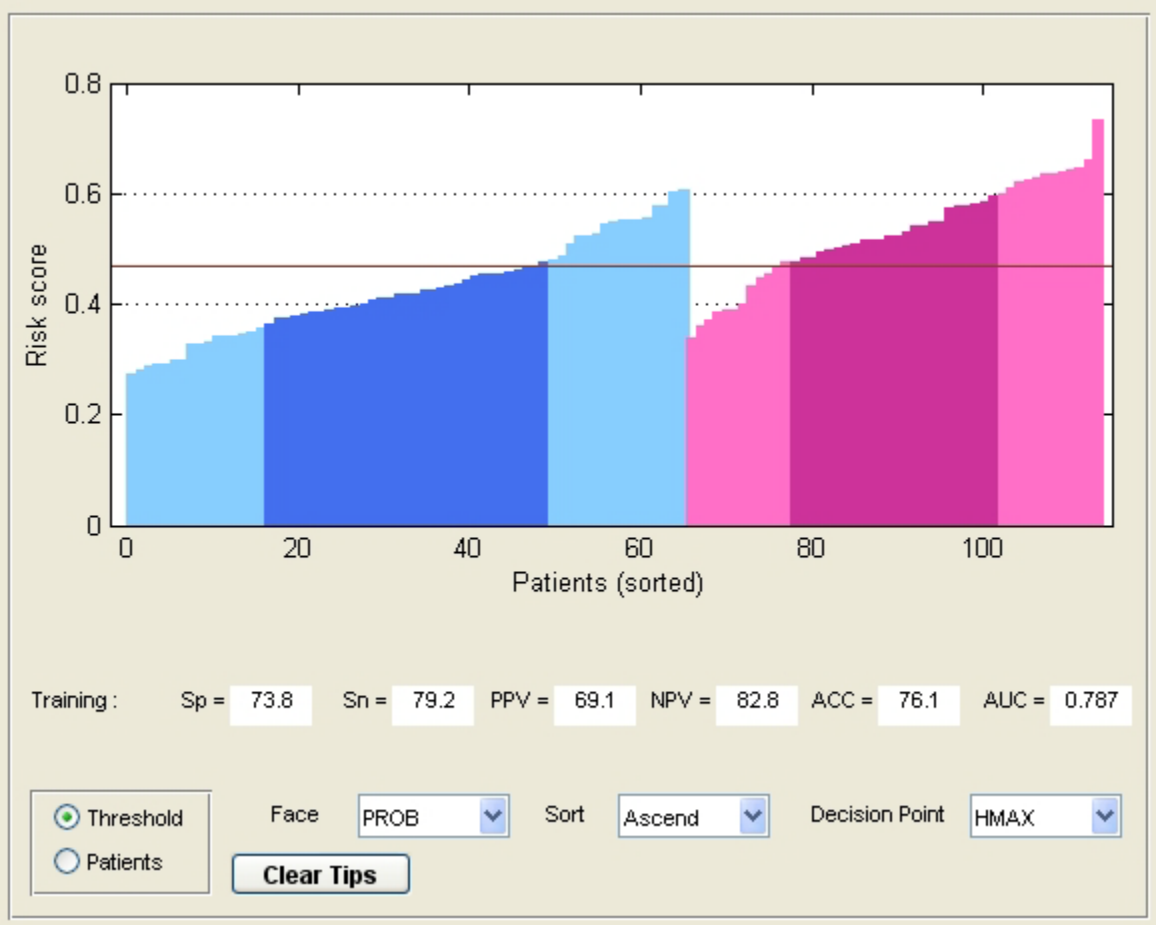

Figure 4.17. Sample IR new plot once plotting is complete.

Clicking the Print button, located in the smaller Plotting controls section, brings up a standard Windows Print Preview dialog box and allows the user to print the completed plot

to a networked printer. The Print Preview dialog box allows the user to stretch or condense the printed plot, as necessary. Once the plotting of data is complete, a larger mirror to the plot of the main axis is displayed in the Plot window of the application. To open the Plot window, the user clicks the Undock button in the Plotting Controls section.

4.9 MAIN WINDOW, STATUS AND ERROR CONTROLS SECTION

The Status and Error Controls section of the GlycoAnalyzer gives the user feedback regarding the status of the GlycoAnalyzer tests. It also allows the user to reset the application, view basic help files, view details on why an error was thrown, and close the application. Figure 4.18 shows the complete Status and Errors controls section of the GlycoAnalyzer application.

Figure 4.18. Status and Error Controls section.

The Status and Error textbox displays messages useful to the user during data processing. Status messages are displayed in black text and error messages are displayed in red text. If a user error is thrown, the issue and possible solution are detailed for the user. If a programming run-time error is thrown, the orange "?" button appears allowing the user to see the filename and line number in the application where the error is thrown.

The Reset button resets the entire GlycoAnalyzer application back to an initial default state. When the Reset button is clicked, a dialog box appears notifying the user that the application is about to be reset. If a reset occurs, all data loaded by the user is erased and each control in the application is reset to a specified initial state.

The Help button displays a help text file. This file lists the function of each control in the application, details about running the application, and the current version of the application. The details listed in the help file are also listed in Appendix A.

The Close button saves the current configuration of the GlycoAnalyzer and any data loaded by the user and exits the application. Before the application closes, a Close dialog appears notifying the user that the application will close. The next time the application is launched, the current configuration is displayed by the application.

4.10 PREPROCESSING WINDOW

The Preprocessing window displays lists of glycans once preprocessing has occurred. Clicking the View Data button in the Preprocessing Controls section opens the Preprocessing window. If preprocessing has not occurred, the Preprocessing window opens in a blank state. Once preprocessing is complete, the labels and glycan numbers are displayed in the open Preprocessing window (see Figure 4.19). The sections that are displayed include (1) glycans used as control spots, (2) glycans that have high correlation, (3) glycans that are rejected due to low intensity, (4) glycans that are rejected due to high CV, (5) glycans that are rejected due to low ICC, (6) list of all rejected glycans.

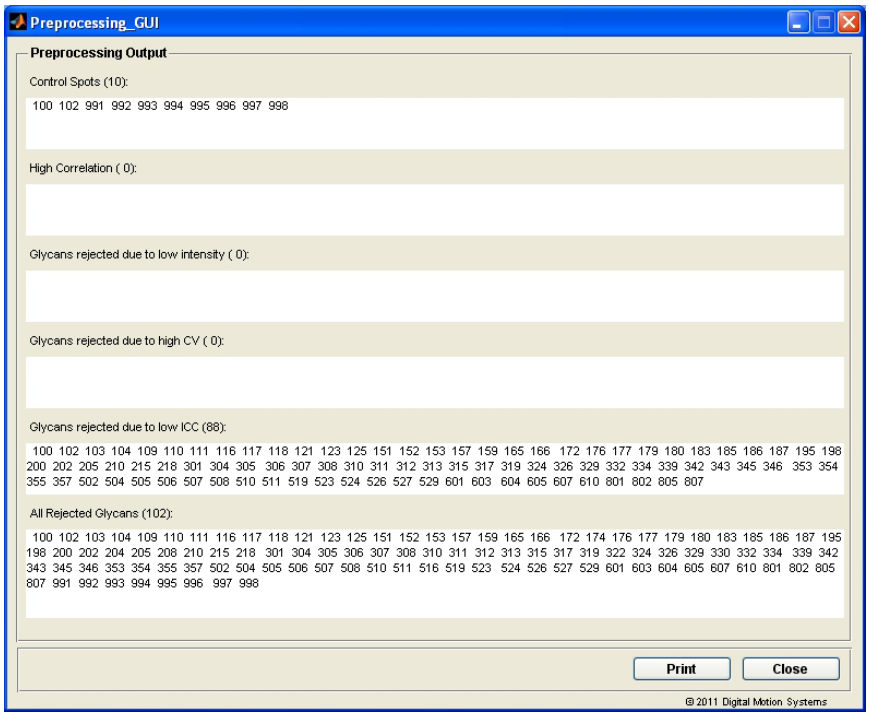

Figure 4.19. Preprocessing window after preprocessing is complete.

Clicking the Close button in the Preprocessing window closes the window. After the window is closed, the results from the current preprocessing run are displayed until preprocessing is run again. Clicking the Print button brings up a standard Windows Print Preview dialog box and allows the user to print a view of the entire Preprocessing window to a networked printer. The Print Preview dialog box allows the user to stretch or condense the printed window, as necessary.

4.11 OUTPUT WINDOW

The Output window displays a list of top-ranked glycans and information about those glycans once the feature selection and projection of data has occurred. Clicking the View Data button in the Feature Selection and Projection Controls section opens the Output window. If feature selection and projection has not occurred, the Output window opens in a blank state. Once feature selection and projection is complete, the labels, top glycan numbers, and information about the top glycans are displayed in the columns of the Output window (see Figure 4.20).

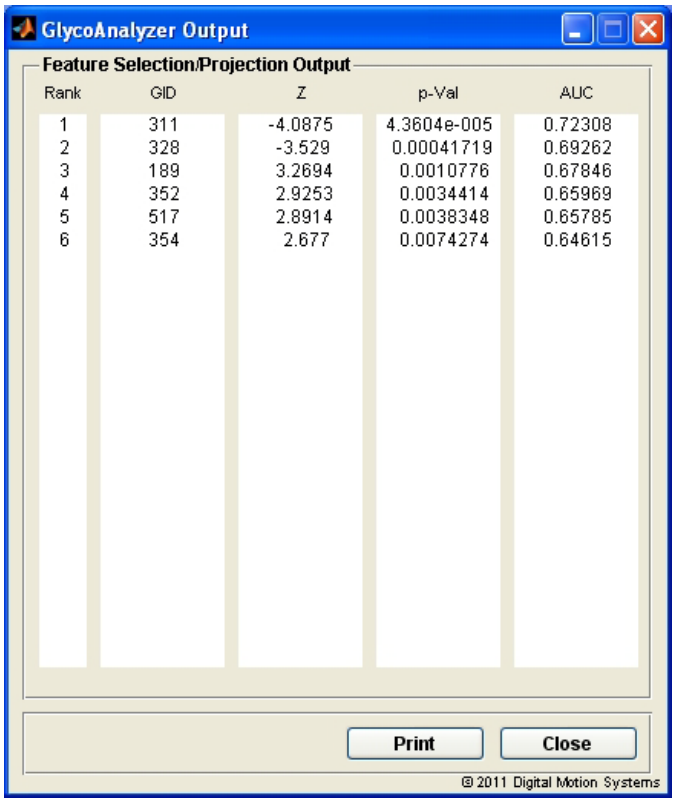

Figure 4.20. Output window after feature selection and projection is complete.

If WMW is selected as the feature selection method, the Output window displays the rank, glycan identification number, Z-value, p-value, and AUC for each top ranked glycan. If any of the other feature selection methods are selected, only the ranking and glycan identification number are displayed in the Output window. The glycan information displayed for each feature selection method will increase during future GlycoAnalyzer updates.

Clicking the Close button in the Output window closes the window. After the window is closed, the results from the current run of data processing are displayed until feature selection and projection is run again. Clicking the Print button brings up a standard Windows Print Preview dialog box and allows the user to print a view of the entire Output window to a networked printer. The Print Preview dialog box allows the user to stretch or condense the printed window, as necessary.

4.12 PLOT WINDOW

The Plot window provides a mirror to the main axis displayed in the Plotting Controls section. Clicking the Undock button in the Plotting Controls section opens the Plot window. If an initial plotting of data on the main axis of the application has not occurred, the Plot window opens in a blank state. Once plotting is complete, an identical plot to the main axis plot will be displayed in the Plot window (see Figure 4.21). The functionality of the plot is the same as that of plot in the Main window of the application.

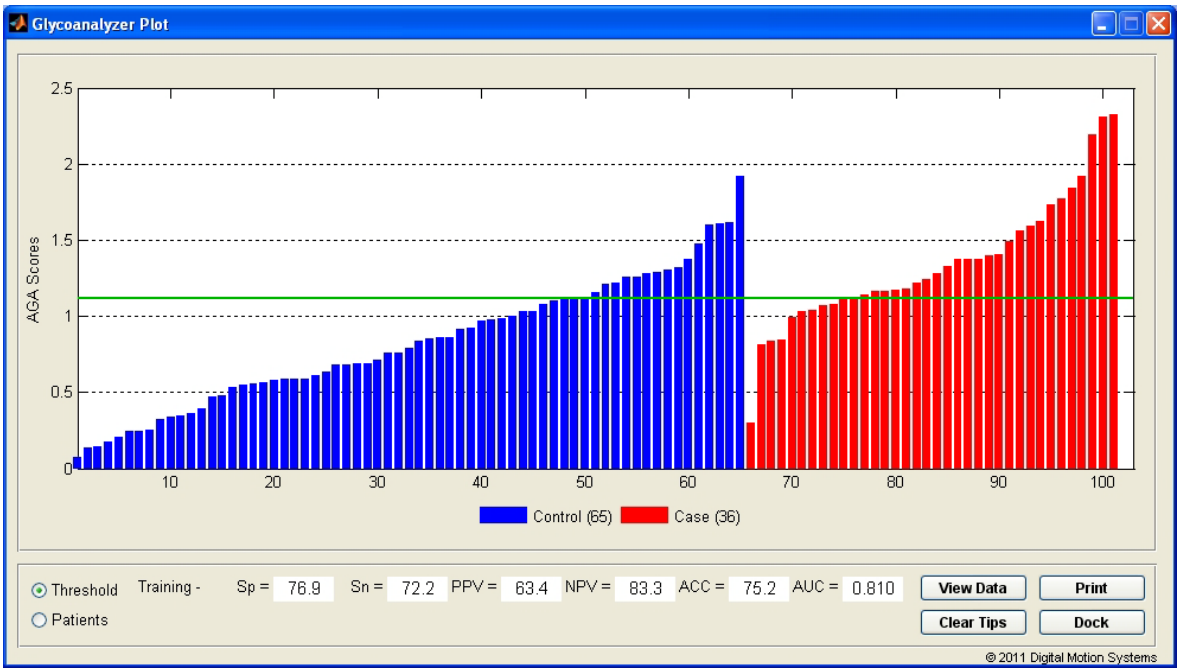

Figure 4.21. Plot window with an example IR plot after plotting is complete.

The Dock button in the Plot window closes the window. After the window is closed, the plot from the current run of data processing is displayed until plotting is run again. The Print button brings up a standard Windows Print Preview dialog box and allows the user to

print a view of the entire Output window to a networked printer. The Print Preview dialog box allows the user to stretch or condense the printed window, as necessary. The Clear Tips button clears any tooltips displaying the patient identification number and risk score. The View Data button opens the Output window and displays information about the top ranked glycans. The Threshold and Patients radio buttons toggles between allowing the user to change the threshold line and selecting the patients to display a tooltip detailing the patient identification number and risk score. Any modification to the Plot window also occurs in the Main window.

CHAPTER 5

IMPLEMENTATION OF THE GLYCOANALYZER IN THE MATLAB GUI ENVIRONMENT

This section specifies how the GlycoAnalyzer application was created and is updated and details how it is launched and used by potential users. Figure 5.1 shows a graphical depiction of the application flow from the design of the application using MATLAB guide.

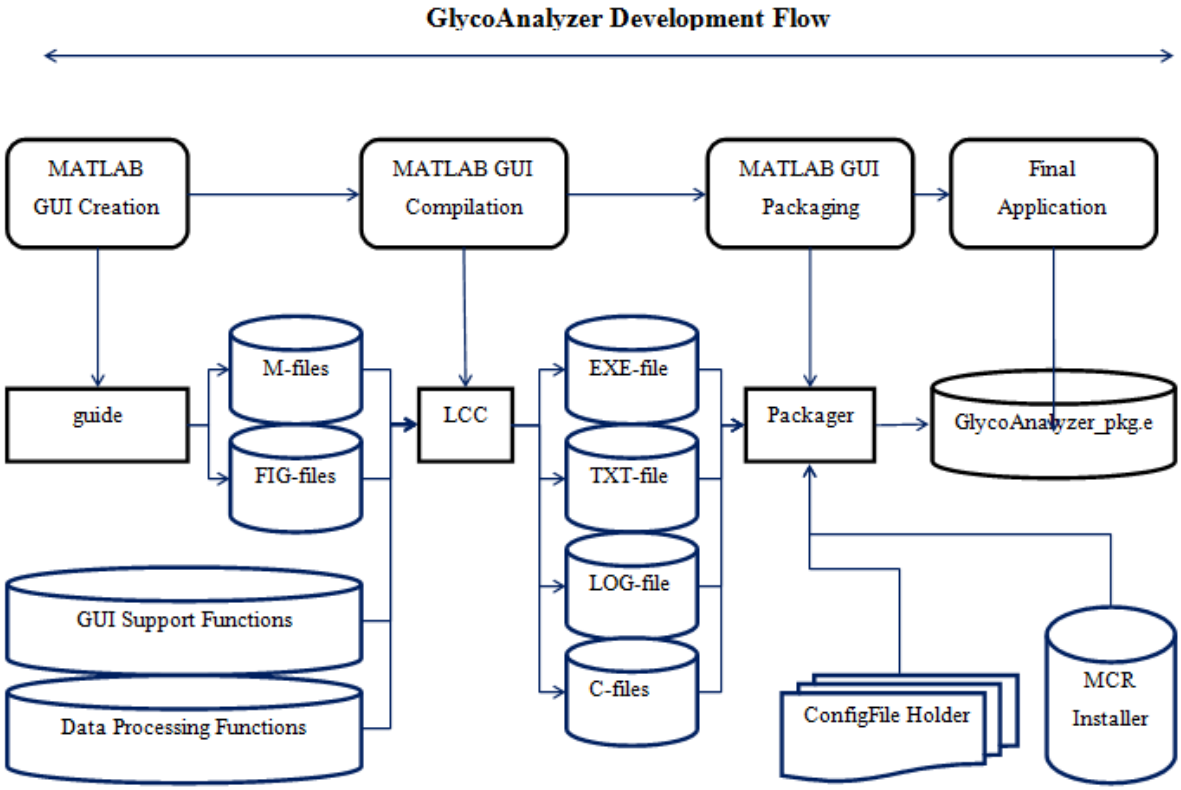

Figure 5.1. Development flow of the GlycoAnalyzer.

Figure 5.2 shows a graphical depiction of the flow of the user when installing and running the application. These diagrams will be discussed, in detail, in this chapter.

The user interface is separated into four main windows: The Main window, the Preprocessing window, the Output window, and the Plot window. The Main window is used to input data files and labels, complete preprocessing, feature selection, and projection, and

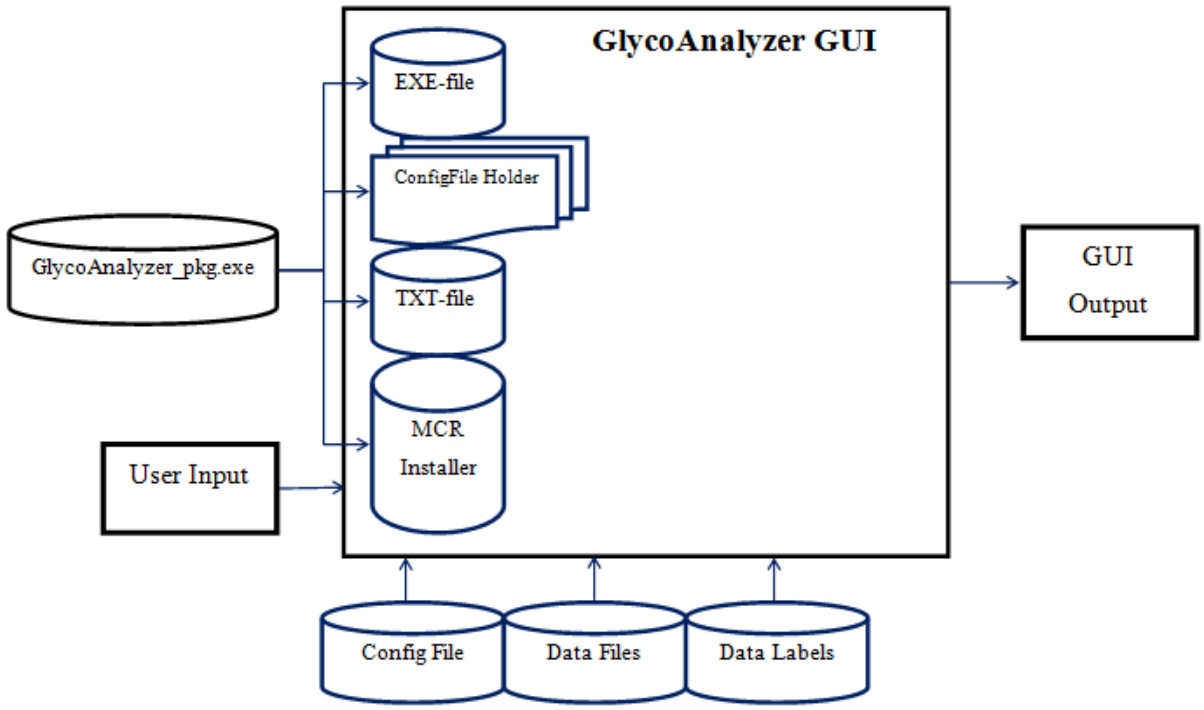

Figure 5.2. User installation and operational flow of the GlycoAnalyzer.

to provide a means for the visualization of data once the processing is complete. The Preprocessing window displays lists of glycans that have been removed once data preprocessing is complete. It also details brief reasons for why each glycan is removed. The Output window contains data related to the top ranked features once feature selection and projection have been completed. Finally, the Plot window is a mirror to the plotted data in the Main window axis and contains the same functionality, but it displays the data in larger axes.

5.1 General Description

The GUI in this project was developed using the MATLAB GUI Layout Editor. MATLAB GUIs can be created completely in code, but the Layout Editor allows the user to drag and drop components onto a blank GUI template, creating the way a GUI looks visually and very quickly. Once the new Layout Editor template is saved, MATLAB automatically creates the required files needed to run any standard MATLAB GUI [36].

To open the GUI Layout Editor, the user types the command, **guide** in the MATLAB Command Window. Implementing **guide** automatically creates a FIG-file and an M-file for

the GUI [37]. The FIG-file is a binary file that holds the complete graphical description of the GUI. This description includes the type, details, and locations of all user interface components, such as push buttons, axes, user interface panels, etc. This FIG-file can only be manually modified using the **guide** command, but additional modification can be done by adding configuration code to the project M-file. The M-file includes code for initializing the GUI and callback functions for controlling each of the GUI components. Once **guide** is implemented from the MATLAB Command Window and the FIG-file is saved, the M-file is created automatically. Several functions and structures are automatically generated for the basic tasks required by any general GUI, including the opening function, the output function, and all of the callback functions required to run the individual components that have been placed on the GUI Layout Editor [38].

Typing **guide** creates an initially blank GUI (see Figure 5.3). Along the left side of the FIG-window, there is a list of available user interface components that can be manually dragged and dropped onto the GUI. Every time a GUI component is added to the FIG-file or modified using the component inspector, callback functions required by the GUI component are automatically added to or modified in the M-file every time the FIG-file is saved. The programmer can then add code to the callback functions that is required to make the component perform specific tasks.

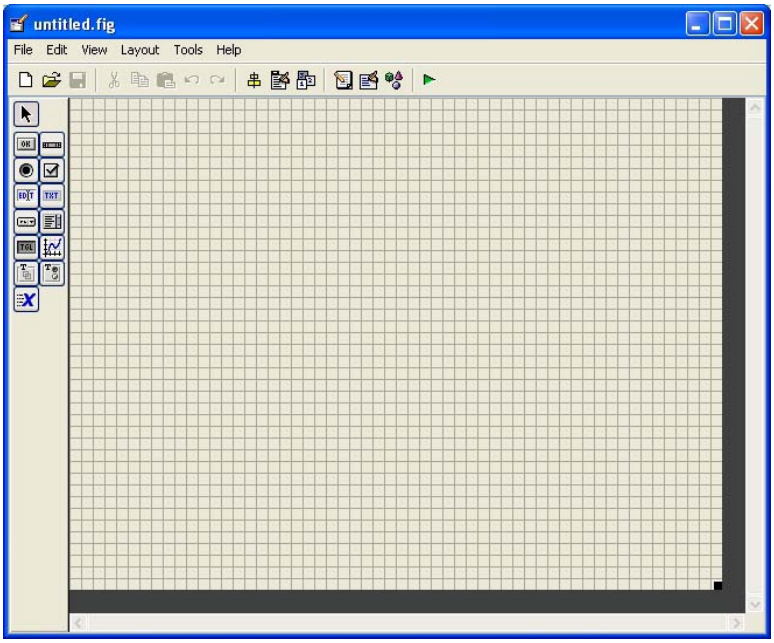

Figure 5.3. Blank MATLAB GUI Layout Editor window.

The GUI used in this project is actually created using four separate GUI windows which have been coded to seamlessly interact with each other using MATLAB handles structures. These structures, while allowing users to call functions, also store data in data structures for later use [39]. The four windows include: the Main window, the Preprocessing window, the Output window, and the Plot window. When the application is launched, the Preprocessing, Output, and Plot window visibility settings are initially set to off in the GUI opening function making the three windows invisible to users. If the Preprocessing and Output window's visibility settings are changed to on once the user clicks the View Data button in each of the respective controls sections of the Main window. The Close button in each of the windows resets the visibility setting to off, hiding each of the windows. The Plot window's visibility settings are changed to on once the user clicks the Undock button in the Plotting Controls section of the Main window. Once the user clicks the Dock button in the Plot window, the visibility settings are once again changed to "Off" and the Plot window is hidden.

Each of the GUI components and figure windows are controlled using MATLAB handles structures. Handles are structures that contain identifiers and details to each of the graphics objects and components specified on the GUI Layout Editor. Every component on the GUI has a list of properties and a handles structure with an identifier is assigned for each object. The root object is given a handle of 0 and each additional component placed on the editor is given a sequential handle so that it can be controlled using code. The available properties for each component vary based on the requirements for the specific component and each of the properties can be referenced in the handles structure [40].

Each of the graphics handles for figures and components can be modified using code or by using the Property Inspector. The Property Inspector contains a complete list of properties for each component. The Property Inspector can be opened by double-clicking a component in the FIG-file. Once opened, it displays a list of available properties for the figure or component. Figure 5.4 displays the Property Inspector for the Feature Selection pop-up menu. The left column of the Property Inspector contains the list of properties and the right column contains the value specified for each column. Right-clicking on any of the values in the right column brings up a menu containing the contents, "What's This?"

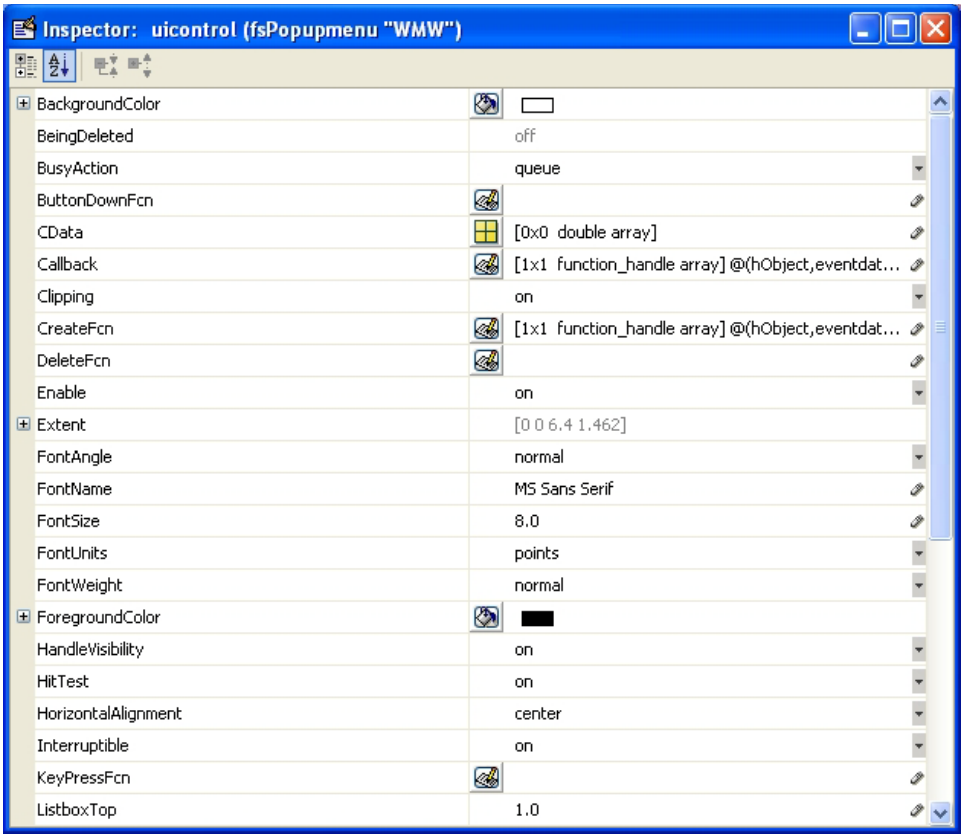

Figure 5.4. Property Inspector for the Feature Selection pop-up menu.

Clicking on the menu item brings up a description of the specified property and available values [41].

The **set** method can be used to modify the component using MATLAB code in the following way:

```
set(hFig_main.statusErrorTxt, 'ForegroundColor','Black');
```

In this example, the handles structure is referenced using **hFig_main** and the component is referenced using the dot operator and the tag property for the component. In this case, the Status and Error textbox is called **statusErrorTxt**. The property to be changed is the **ForeGroundColor** and the value for it to change to is **Black** [42].

5.2 Support Functions

The GlycoAnalyzer contains two distinct sets of functions. The first set is the group of functions that Vuskovic has created and perform the calculations required for preprocessing, feature selection, projection, and plotting. The second set of functions

contains the support functions required to run the GUI. These files are designed to control the GUI from opening to closing and perform other administrative tasks, such as (1) loading and deleting data, (2) error checking, (3) controlling the visibility of axes, (4) disabling and enabling components, (4) resetting the GUI component values, (5) setting and getting values related to GUI functions, (6) saving and retrieving values to and from the GUI handles functions. The support functions have been separated into their own GUI subdirectory in Vuskovic's files and each has been given a _GUI name to designate them as GUI specific functions. Figure 5.5 details the interaction of the GlycoAnalyzer with the different types of functions used by the application.

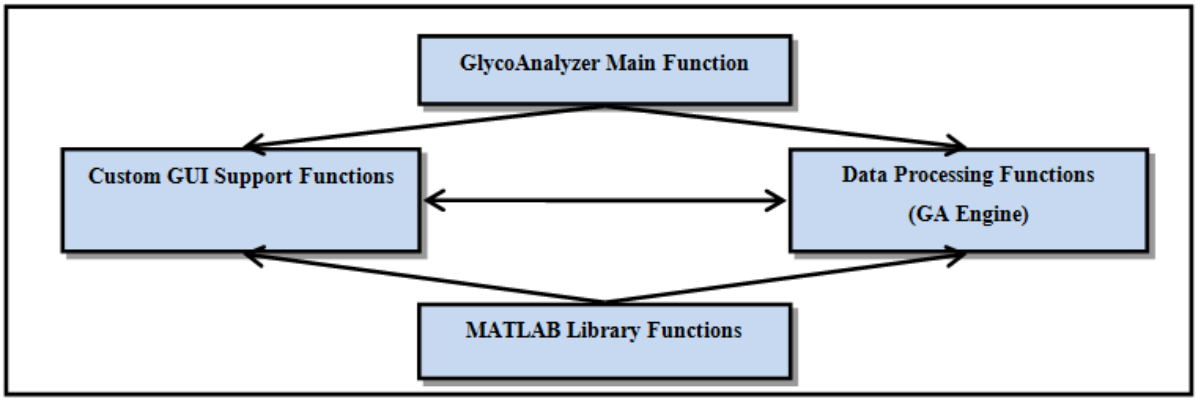

Figure 5.5. Diagram of GlycoAnalyzer function structure.

5.3 Structure of the MATLAB GUI Run-Time System

When the GlycoAnalyzer application is compiled into a standalone executable file, that executable consists of a combination of C and MATLAB files that integrate to form the final application for end-users. The application could have been completely written using the C or C++ languages, but MATLAB includes standard libraries that make mathematical calculations easier and more efficient. Normally, MATLAB M-files can only be run within MATLAB development environment. Fortunately, the full version of MATLAB includes a built in compiler and compiler toolbox that allows MATLAB projects to be compiled into EXE-file applications and run on any workstation. This allows developers to easily distribute applications written in the MATLAB environment to end-users [43].

During the compilation process, two directories and several files are created in the project specified folder: src and distrib. The src directory holds the files required to run the compiled executable application outside of the MATLAB environment. These files form a wrapper and integrate directly with the M-files from the project. The src directory also holds the compiled executable file and log files from the compilation process. Table 5.1 describes the main files that are created during the project compilation [44]. The distrib folder contains the compiled component file that can be installed as a standalone executable on end-user PCs.

Table 5.1. Files Created During Compilation

File Name	Purpoes
GlycoAnalyzer_main.c	Contains the C-code main function for the application. This file provides a wrapper for the MATLAB code and allows input arguments usually passed on the command line to to be passed to the GlycoAnalyzer application.
GlycoAnalyzer_mcc_component_data.c	Contains the C-code needed by the MATLAB Compiler Runtime (MCR) to run the application and specifies the paths, encryption keys, and formatting required for the MCR. The MCR includes platform specific libraries required to run M-files.
GlycoAnalyzer.exe	The main file of the GlycoAnalyzer application. This file uses the files stored in the CTF-archive to run the compiled application. The CTF-archive stores the M-files that are imported during the compilation process.

Once the application is fully compiled, the GlycoAnalyzer is ready for the packaging stage. During packaging, a self-extracting executable is created that contains the application executable file along with any supporting files required for the application to run. In this case, the ConfigFileHolder directory and possibly the MCR Installer file are included in the packaged executable file. The ConfigFileHolder folder contains the global variables file, the GlycoAnalyzer configuration file, a test data MAT-file containing patient date, and the data labels XLS-file that works with the test data file. A complete list of the global variables can be found in Appendix B.

If the GlycoAnalyzer will be installed for the first time on a new system, the MATLAB Compiler Runtime (MCR) Installer must be included in the packaged in the component installer created by the packaging process. The MCR Installer contains libraries that allow users to run MATLAB files on PCs even if MATLAB is not installed on that PC. The MCR Installer only needs to be run once on each PC. Once it is installed, it does not have to be included with each successive version of the packaged application. If the MCR Installer is packaged with the GlycoAnalyzer, the user will be prompted to install it automatically when the GlycoAnalyzer is run for the first time. It can be installed in the default location [45].

5.4 Compiling MATLAB Code and Building the Stand-Alone Application

Ultimately, the goal of the GlycoAnalyzer project was to develop an application that could be installed on any other PC running a Microsoft Windows operating system, even if that PC did not have a copy of MATLAB installed. Fortunately, the full version of MATLAB comes equipped with a built in C++ compiler, called Lcc, which is able to translate MATLAB M-files into C++ code. In addition, MATLAB 2010a also supports other 32-bit C++ compilers, including the Microsoft Visual C++ 10.0, Microsoft Visual C++ 9.0, Microsoft Visual C++ 8.0, Microsoft Visual C++ 6.0, Intel C++ 11.1 and Open Watcom 1.8 compilers [46]. The executable that is created, after compilation, can be run on any PC, provided the PC is running the same OS as the PC that created the executable. An executable file created on a PC running XP, can also be run on PCs running Vista and Win7.

5.4.1 Locating and Setting-up the Installed and Supported Compilers

The first step in compiling a MATLAB application is to locate and setup the installed, supported compilers. To do this, the following steps must occur:

1. In the MATLAB Command Window, type the command: `mbuild –setup`
2. When the question, “Would you like mbuild to locate installed compilers?” appears, type “Y” and press ENTER.
3. When the list of installed and supported compilers appears, type the number of the desired compiler and press ENTER.

4. MATLAB will ask the user to verify the choice of compilers. If your choice was correct, type "Y" and press ENTER.

At this point, the newly selected compiler is the default compiler used each time the MATLAB project is complied. These instructions can be used each time a new compiler is desired.

5.4.2 Deploying the GlycoAnalyzer to End-Users

In order for the GlycoAnalyzer application to be easily used by a variety of end-users, it must be compiled and packaged into a stand-alone executable file. The Deployment Tool, built into the full version of MATLAB, is used to do this. The Deployment Tool is launched by typing the command `deploytool` in the MATLAB Command Window. This launches the Deployment Tool user interface in a sub-window within the MATLAB Command Window [47]. The Deployment Tool user interface allows programmers to build an application using installed C++ compilers and package the application into a single executable file for end users. This EXE-file can be configured to include all of the MATLAB code, the MATLAB MCR Installer, and any files required by the application to run. Double-clicking on the EXE-file unpackages it on the end-user's PC.

5.4.2.1 Building a New GlycoAnalyzer Deployment Project

Once the Deployment Tool user interface is open in MATLAB, it can be used to create a new packaged application. The steps listed here follow the steps for creating and packaging an application listed in the Magic Square Example [48]. The steps to do this, written with the GlycoAnalyzer application in mind, are as follows:

1. Create a subdirectory in the GlycoAnalyzer directory and call it GlycoAnalyzer. On my PC, this subdirectory is located in: C:\THESIS\GUI\GlycoAnalyzer\.
2. If it is not already open, in the MATLAB Command Window, type `deploytool` to open the Deployment Project dialog box.
3. In the Deployment Project dialog box, click the New tab.
4. Type GlycoAnalyzer.prj in the Name textbox.
5. Click the Browse button to the right of the Location textbox and browse for the GlycoAnalyzer folder created in Step 1.
6. Select Console Application in the Target pop-up menu.

7. Click the OK button in the Deployment Project dialog box to create the project. This will create the new GlycoAnalyzer package project in the Deployment Tool user interface. The project now contains two empty sections: Main File and Shared Resources and Helper Files.
8. Click on the Build tab at the top of the Deployment Tool user interface.
9. Add the main file by clicking the Add Main File link in the Main File section of the Deployment Tool user interface. Browse for the file: Immunoruler_GUI.m in the Windows Add File dialog box and add it to the project by clicking the Open button. This is the main file for the GlycoAnalyzer application.
10. Add the each of the supporting files by clicking the Add Files/Directories link in the Shared Resources and Helper Files section of the Deployment Tool user interface. All M-files and FIG-files used by the application must be added. Browse for each of the supporting files using the Windows Add File dialog box and add them to the project by clicking the Open button. Multiple files can be added at once by CNTL-clicking each file in the Add File dialog box.
11. Click the Build icon in the Deployment Tool toolbar to compile and build the project.

As the GlycoAnalyzer application is built, two directories and several files are placed in the GlycoAnalyzer folder that was created in Step 1. These directories are (1) src (2) distrib. The files placed in the distrib directory include (1) _install.bat, (2) GlycoAnalyzer.exe, (3) readme.txt. The files placed in the src directory include:

1. build.log
2. GlycoAnalyzer.exe
3. GlycoAnalyzer _delay_load.c
4. GlycoAnalyzer _main.c
5. GlycoAnalyzer_mcc_component_data.c
6. mccExcludedFiles.log
7. readme.txt. The file, GlycoAnalyzer.prj, is also created during this process.

5.4.2.2 BUILDING AN EXISTING GLYCOANALYZER DEPLOYMENT PROJECT

Once the initial GlycoAnalyzer deployment package has been completed, it can be easily modified or rebuilt, as needed. The steps for doing this are:

1. If it is not already open, in the MATLAB Command Window, type `deploytool` to open the Deployment Project dialog box.
2. In the Deployment Project dialog box, click the Open tab.

3. Navigate to the GlycoAnalyzer.prj file by clicking the Browse button. Click the Open button to open the project.
4. Click the OK button in the Deployment Project dialog box to load the GlycoAnalyzer project file.
5. Add any new supporting files to the files to the Add Files/Directories link in the Shared Resources and Helper Files section of the Deployment Tool user interface. All existing files, including files that have been modified are already saved in the project. Only new supporting files must be added to the project during this step.
6. Click the Build icon in the Deployment Tool toolbar to compile and build the project.

5.4.2.3 PACKAGING THE GLYCOANALYZER APPLICATION FOR DEPLOYMENT

Packaging the GlycoAnalyzer allows users to copy a single GlycoAnalyzer EXE-file into a specified location, running the application easily from that location. Once the application has been built using the previous steps, packaging the application creates the single executable file. Once the Deployment Tool user interface is open in MATLAB, it can be used to create a new packaged application from the previously compiled application. The steps to do this are as follows:

1. If it is not already open, in the MATLAB Command Window, type `deploytool` to open the Deployment Project dialog box.
2. In the Deployment Project dialog box, click the Open tab.
3. Navigate to the GlycoAnalyzer.prj file by clicking the Browse button. Click the Open button to open the project.
4. Click the OK button in the Deployment Project dialog box to load the GlycoAnalyzer project file.
5. Click on the Package tab at the top of the Deployment Tool user interface.
6. Add the MonkeyHolder directory to the project by clicking Add Files/Directories link and browsing the the MonkeyHolder directory using the Windows Add Files dialog box. Click the Open button to add the directory to the package.
7. If this GlycoAnalyzer package will be installed for the first time on a particular PC, click the Add MCR link to add the MCR Installer file to the package. The MCR Installer includes all of the necessary files required to run packaged MATLAB projects on user PCs. Once the MCR Installer has been installed on a particular PC, it can be removed from the project to save space.
8. Click the Package icon in the Deployment Tool toolbar to package the GlycoAnalyzer project. When the GlycoAnalyzer project is packaged, the GlycoAnalyzer_pkg.exe file is created and placed in the project directory.

5.4.2.4 Deploying the GlycoAnalyzer Application to End-Users

Once the GlycoAnalyzer application has been successfully built and packaged, it can be sent to end users as a single EXE-file. Packaging the application has two main benefits. First, it allows the user to copy a single EXE-file rather than the entire application folder that is created during the compilation and building phase. Second, it hides the application code from end-users, preventing the application from being recreated by other developers.

If it is the first time the application has been run on a user's PC, the MATLAB Compiler Runtime (MCR) application must be part of the package and installed on the user's PC prior to being able to run the GlycoAnalyzer. Once the MCR has been installed, the application can be built and packaged without the MCR, reducing the size of the overall application and the time required for installation. The steps to installing the GlycoAnalyzer on an end-user's PC is as follows:

1. Create a subdirectory on the user's PC called: C:\GlycoAnalyzer\. If the subdirectory is already created, delete all files and folders in the subdirectory.
2. Copy and Paste the file GlycoAnalyzer_pkg.exe into the GlycoAnalyzer subdirectory.
3. Double-click on the GlycoAnalyzer_pkg.exe file to unpack the application. Running this file will copy files to the subdirectory including (1) MCRInstaller.exe (2) MonkeyHolder folder (3) GlycoAnalyzer.exe (4) readme.txt.
4. If this is the first time the application is run, the prompt to install the MCR will appear automatically. Follow the prompts and install the MCR in the default location.
5. Double-click on the GlycoAnalyzer.exe file to run the GlycoAnalyzer application normally from the GlycoAnalyzer subdirectory.

5.5 General Application Update

This section describes the process for updating the GlycoAnalyzer application, including (1) updating any existing functions, (2) adding new functions, (3) adding new components, (4) adding new windows. The code in many of the GlycoAnalyzer functions is constantly being updated and improved by Dr. Vuskovic and his associates. Each time a file used by the GlycoAnalyzer is updated, it must be checked to ensure it will work correctly with the GlycoAnalyzer application. In order for this to happen, the following items must be checked:

1. The global GUI handles structure, hFig_main must be added to the file if the file is to interact with any of the GUI components.
2. Any use of the MATLAB function, error, must be replaced by the custom function, My_error. This allows the error output to be properly displayed in the GUI Status/Error textbox.
3. Any use of the MATLAB functions, close, must be replaced by the custom function, My_close. This prevents the GUI figure windows to be prematurely terminated while the user is running the GlycoAnalyzer application.
4. If the application will be compiled as a Windows Standalone Application, any text output to the MATLAB Command Window must be suppressed using the function, My_disp. Windows Standalone Applications will crash if any text is output to the Command Window. The function My_disp prevents text output.

5.5.1 Updating Existing Functions in the GlycoAnalyzer Application

The following steps can be used to update any existing function in the GlycoAnalyzer application:

1. In MATLAB, open the existing function that will be modified.
2. Update the code in the function, following all of the steps in section 4.5.1.
3. Once the changes are complete, compile the application using the instructions listed in section 5.4.2.2.
4. Package the application using the instructions listed in section 5.4.2.3.

5.5.2 Adding New Files to the GlycoAnalyzer Application

When a new file is needed in the GlycoAnalyzer application, adding the new files is relatively easy. New files may be required to add future functionality to the GUI, such as adding new feature selection methods like the Ant Colony or Random Forest algorithms. New files may also be used to create easier-to-read code. To add a new file, the following steps need to occur:

1. Create a new M-File by clicking the New M-File icon in the MATLAB toolbar. Make sure that the code will work seamlessly with the GUI using the steps listed in section 4.5.1.
2. Add the file to the GlycoAnalyzer project file using the steps listed in section 5.4.2.2.
3. Compile the application using the instructions listed in section 5.4.2.2.
4. Package the application using the instructions listed in section 5.4.2.3.

5.5.3 Deleting Files from the GlycoAnalyzer Application

When a file is no longer needed in the GlycoAnalyzer application, delete the file is using the following steps:

1. Remove any reference to the file from all of the other files in the application.
2. If it is not already open, in the MATLAB Command Window, type **deploytool** to open the Deployment Project dialog box.
3. In the Deployment Project dialog box, click the Open tab.
4. Navigate to the GlycoAnalyzer.prj file by clicking the Browse button. Click the Open button to open the project.
5. Click the OK button in the Deployment Project dialog box to load the GlycoAnalyzer project file.
6. In the GlycoAnalyzer deployment project, click on the Build tab.
7. In the Shared Resources and Helper Files section, right-click on the file to be deleted and click the Remove from the menu.
8. Compile the application using the instructions listed in section 5.4.2.2.
9. Package the application using the instructions listed in section 5.4.2.3.

5.5.4 Adding Components to the GlycoAnalyzer Application

As the functionality of the GlycoAnalyzer increases, often new GUI components need to be added to the application FIG-files. Adding new components can be completed using the following steps:

1. In the MATLAB Command Window, type the command **guide** to open the MATLAB GUI Layout Editor.
2. Click on the Open Existing GUI tab in the GUIDE Quick Start dialog box.
3. Navigate to the desired FIG-file and click the Open button to open the FIG-file in the GUI Layout Editor.
4. Drag and drop the desired components onto the FIG-file, arranging them with the existing components. The Align Objects feature helps align the new components with existing components once they have been placed in the FIG-file.
5. Click the Save Figure button in the GUI Layout Editor toolbar to create the callback functions required to operate the new component. The callback functions will appear automatically in the M-file associated with the FIG-file.
6. Open the M-file associated with the FIG-file and find the newly created callback functions.

7. Add code to the callback function to make the component work correctly with the rest of the GUI.
8. Once the code is complete, click the Save button in the M-file Editor toolbar.
9. Compile the application using the instructions listed in section 5.4.2.2.
10. Package the application using the instructions listed in section 5.4.2.3.

5.5.5 Deleting Components from the GlycoAnalyzer Application

When a GUI component becomes obsolete or the functionality is changed and uses a different type of component, the old GUI component should be promptly removed from the FIG-file associated with the component. The M-file containing the component's callback function should also be modified so that the callback function no longer exists. Deleting unused components will reduce confusion as the GUI is modified by different programmers. Deleting components from the GlycoAnalyzer application can be completed using the following steps:

1. In the MATLAB Command Window, type the command **guide** to open the MATLAB GUI Layout Editor.
2. Click on the Open Existing GUI tab in the GUIDE Quick Start dialog box.
3. Navigate to the desired FIG-file and click the Open button to open the FIG-file in the GUI Layout Editor.
4. Select the GUI component to be deleted and press the Delete button to remove the component from the Fig-file.
5. Click the Save Figure button in the GUI Layout Editor toolbar.
6. Open the M-file associated with the FIG-file and navigate to the callback functions for the deleted component.
7. Delete the callback functions for the deleted component.
8. Once the callback function is removed, click the Save button in the M-file Editor toolbar.
9. Compile the application using the instructions listed in section 5.4.2.2.
10. Package the application using the instructions listed in section 5.4.2.3.

5.5.6 Adding Auxiliary Windows to the GlycoAnalyzer Application

The Preprocessing, Output, and Plot windows all required the addition of a new window to the GlycoAnalyzer application. Each window was designed to integrate

seamlessly with the original GlycoAnalyzer application. To add additional windows to the GlycoAnalyzer application, the following steps must occur:

1. In the MATLAB Command Window, type **guide** to open the MATLAB GUI Layout Editor.
2. From the GUI Quick Start dialog box, select the Create New GUI tab.
3. From the list of default GUIs, select the Blank GUI item.
4. Check the Save the New Figure As: checkbox and name the new window.
5. Click the OK button to open the GUI Layout Editor, displaying a blank GUI canvas.
6. Double-click on the untitled figure to open the Matlab FIG-file Inspector.
7. Set the Name property of the new figure window. Use a name that relates to the functionality of the new window.
8. Set the Visibility property of the new figure window to Invisible.
9. Place all of the required components on the blank figure window and click the Save button to create the M-file for the new window and all of the callback functions for the added components.
10. In the MATLAB Command Window, type **guide** to open the MATLAB GUI Layout Editor a second time.
11. From the GUI Quick Start dialog box, select the Choose Existing GUI tab.
12. Browse for the file Immunoruler_GUI.fig and click the Open button to open the FIG-file.
13. Add any components required to interact with the new GUI and click the Save button to create the callback functions for the new components.
14. Open the file, Immunoruler_GUI.m.
15. In the file, Immunoruler_GUI.m, navigate to the function, `Immunoruler_GUI_OpeningFcn()`
16. Add a new global handles structure for the new GUI, naming the new structure appropriately.
17. Set the visibility of the new GUI to invisible with the code:

```
a. eval('NewGUIName_GUI')
b. set(hFig_NewHandlesStructure.newFigureName,
c. 'Visible','Off');
```

18. Navigate to newly created component callback functions created in step 13 and add code to interact with the new window. This includes changing the visibility of the new window to on.
19. Save the file Immunoruler_GUI.m.

20. Open the M-file created for the new GUI figure window.
21. Add the new window global handles structure to the function, `Output_GUI_OpeningFcn()`.
22. Add code to each of the callback functions to make the components operate correctly.
23. Save the M-file for the new GUI.
24. Compile the application using the steps listed in section 5.4.2.2. The new M-file and FIG-file both need to be added to the Glycoanalyzer project's Shared Resources and Helper Files folder.
25. Package the application using the steps listed in section 5.4.2.3.

5.5.7 Deleting Auxiliary Windows from the GlycoAnalyzer Application

When an auxiliary window is no longer needed in the GlycoAnalyzer application, the M-file and FIG-file for the window should be removed from the project, as well as any reference to those files in the application. The instructions for removing auxiliary windows from the GlycoAnalyzer application are as follows:

1. Delete all references to the window from the file, Immunoruler_GUI.m
2. Delete all components required for the window from the file, Immunoruler_GUI.fig.
3. If the window may be used again in the future, move the window's FIG-file and M-file from the location: C:\THESIS\GUI\ to a new location outside of the project. If the window will never be used again, they can both be deleted.
4. Compile the application using the steps listed in section 5.4.2.2. The auxiliary GUIs M-file and FIG-file need to both be removed from the GlycoAnalyzer project's Shared Resources and Helper Files folder.
5. Package the application using the steps listed in section 5.4.2.3.

5.6 Implementation Issues

The GlycoAnalyzer application represents a significant step forward in the processing of PGA data. Prior to the creation of the GUI, the processing of printed glycan array data was completed by loading the data into the MATLAB Workspace and calling each function manually from the MATLAB Command Window. The GUI simplifies this process by allowing users to manipulate data using specific MATLAB GUI components, such as pop-up menus, push buttons, checkboxes and editable textboxes. Once the printed glycan array data is loaded into GlycoAnalyzer GUI by the user, much of the actual data manipulation is done

by functions that were created over the past few years by Dr. Vuskovic and his associates. Creating the GlycoAnalyzer GUI from previously created files brings a set of unique challenges because each file needs to be checked to make sure it is integrated properly in the GUI environment.

First, in order to create an executable application that can be run on any Windows PC, all of the functions used by the GlycoAnalyzer GUI must be listed in the MATLAB deployment project file when the application is compiled. Some of the files were selected from a group of hundreds of application library functions. The remainder of the application files was created specifically for the project. During compilation, if any of the required files are left out, they will not be available in the running executable, possibly causing the application to crash or have reduced functionality when it is run by end-users.

This issue was fixed by keeping an accurate list of files during the application development. The files that were created specifically for the GUI were kept in a single folder away from the functional files used by the GUI. This made them easy to find and add to the deployment project. The application library files were added to the deployment project from the running list of required files. Once compiled, the application functionality was tested thoroughly for errors thrown because of missing files. Each time an error was thrown because of a missing file, that file was added to the list and added to the Shared Resources and Helper Files folder in the deployment project. The complete list of files required by the GlycoAnalyzer application can be found in Appendix C.

Second, the GlycoAnalyzer data processing engine files are constantly being updated and changed on a regular basis by its developers. Originally, each file used by the GlycoAnalyzer was separated from the original directory of files, copied into a separate folder, and given a modified name to distinguish that file from the original file and allow for changes required for GUI functionality. This method was not acceptable because the original files are constantly changing and being optimized, making the files used by the GlycoAnalyzer quickly obsolete. In addition, updating each modified GUI file individually once the original file was changed became labor intensive and was not efficient.

This issue was fixed when Vuskovic specified that he wanted his original files to be compiled for the application in their original directory instead of separated into a GUI-specific directory. This meant that the original files had to work, both with the GUI, and

separately from the MATLAB Command Line. There were several changes that had to occur for each file in order for this to happen. The specific changes include:

1. Any use of the MATLAB function, **close**, had to be suppressed for the GUI. The function, **close**, causes the GUI exit, making the GlycoAnalyzer application unusable. A new function, My_close.m was created to suppress the use of **close** function so that when the GlycoAnalyzer is running it will not exit uncontrollably (see Figure 5.6).

```
function My_close()

    global GUI_flag

    if ~GUI_flag
        close all;
    end
end
```

Figure 5.6. Function: My_close.

2. Any use of the MATLAB function, **error**, had to be changed so that it would properly output the issue to the GUI Status/Error textbox each time an error was thrown. A new function, My_error.m, was created so that any time the **error** function was called, the GlycoAnalyzer would properly display the error for the user (see Figure 5.7).

```
function My_error(varargin)

    global hFig_main
    global GUI_flag

    msg = varargin(1);

    if GUI_flag
        beep;
        [ST,I] = dbstack(1);
        hFig_main.ST = ST;
        set(hFig_main.genError,'Visible','On');
        set(hFig_main.statusErrorTxt,'ForegroundColor','Red',...
            'String',[' SYSTEM ERROR: Click the ''?'' button for '...
            'details about the error.']);
        error(msg, varargin(2:end));
    else
        error(msg, varargin(2:end));
    end
end
```

Figure 5.7. Function: My_error.

3. Any output to the MATLAB Command Window needed to be suppressed so the application could be compiled as a Windows Standalone Application. A Windows Standalone Application prevents the Windows Command Prompt from running alongside the GlycoAnalyzer application. This makes the application more user-friendly and less confusing. Without the Command Prompt, any display output from the application using the command, **disp**, or **fprintf**, causes the GUI to crash. A new function, My_disp.m, was created to suppress any display output to the MATLAB Command Prompt.

Each of the items listed above were implemented using a new global variable, **GUI_flag**. This variable is set, automatically, when the GUI is launched. If the variable is set, the three functions assume the GUI is being used. If it is not set, the functions can be used outside of the GUI in the MATLAB Command Prompt.

Finally, if errors were thrown while the GUI is running, there is no indication in which file the error was thrown making debugging the run-time error difficult. To fix this issue, a new feature was added to the function My_error.m making error tracing much easier. When a programming error is thrown while the GUI is running, the message from the error is automatically displayed in the Status/Error textbox in the application. In addition, an orange "?" button appears which uses the stack trace to detail where the error was thrown. When the user clicks the "?" button, a dialog box appears detailing the exact file and line number of the error. From there, the user can contact support to have the issue resolved.

CHAPTER 6

RESULTS

This section details a typical use case scenario for the GlycoAnalyzer application from opening the application in an initial state until the plotting of data once preprocessing, feature selection, and projection is complete. This use case details the typical flow of operations through the application.

When the GlycoAnalyzer is opened for the first time, the application components are set in their initial state and the browse button next to the Load Training Data section is highlighted in red, indicating that loading the training data is the first step for the user (see Figure 6.1).

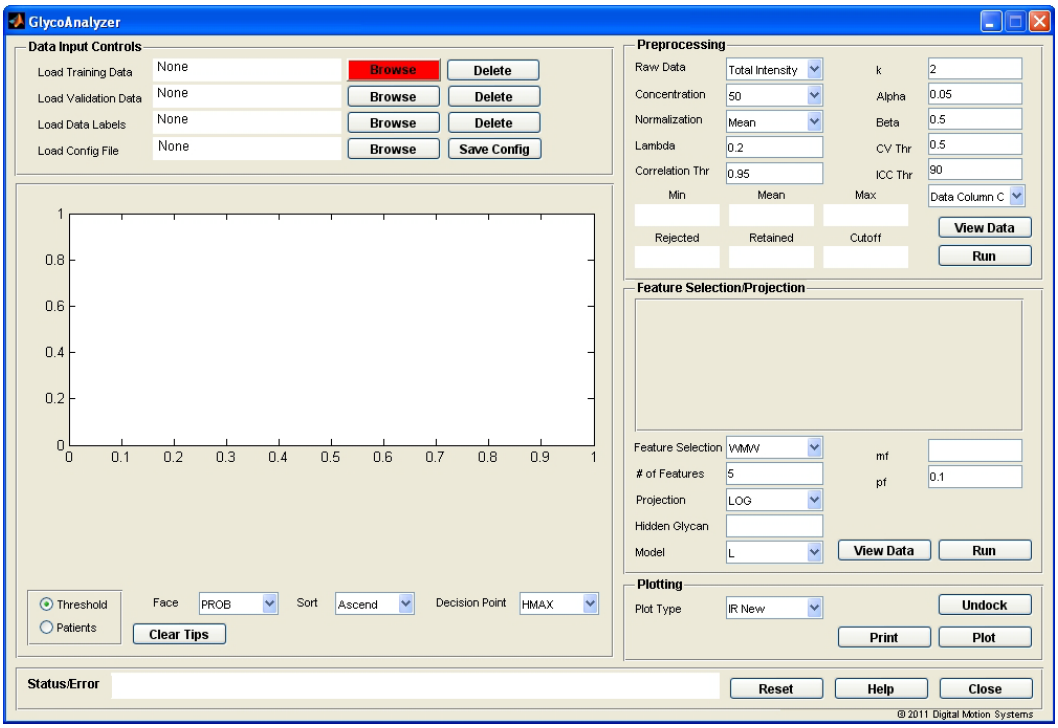

Figure 6.1. Open GlycoAnalyzer application in an initial state.

Loading the training data file and data labels files occur in the Data Input Controls section of the application. To load the training data file, click on the red Browse button next

to the Load Training Data section and browse for a properly formatted data MAT-file using the standard windows Search dialog box (see Figure 6.2). In this study, the training data is from a Mesothelioma study. The selected file is named *Meso.mat*.

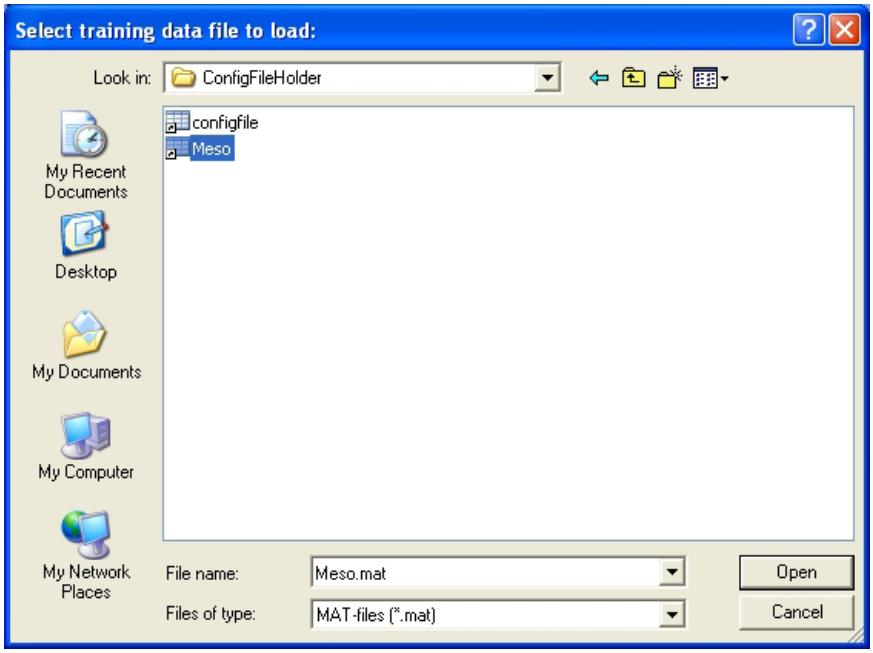

Figure 6.2. Training Data Search dialog box.

Once the training data is loaded, the name of the file is displayed in the Load Training Data textbox and the Browse button next to the Load Data Labels section is highlighted in red, signaling the next step in the application (see Figure 6.3).

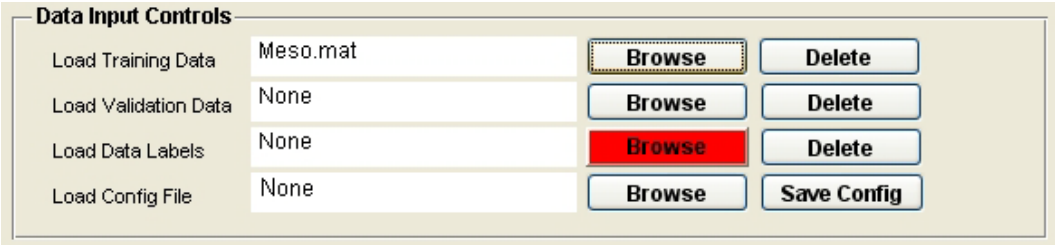

Figure 6.3. Data Input Controls section after the training data is loaded.

Load the data labels file by clicking on the red Browse button next to the Load Data Labels section. Again, use the standard windows Search dialog box to browse for a correctly formatted XLS-file containing the data labels for the current study. In this case, the data labels file for the Mesothelioma study is called Meso_labels.xls (see Figure 6.4).

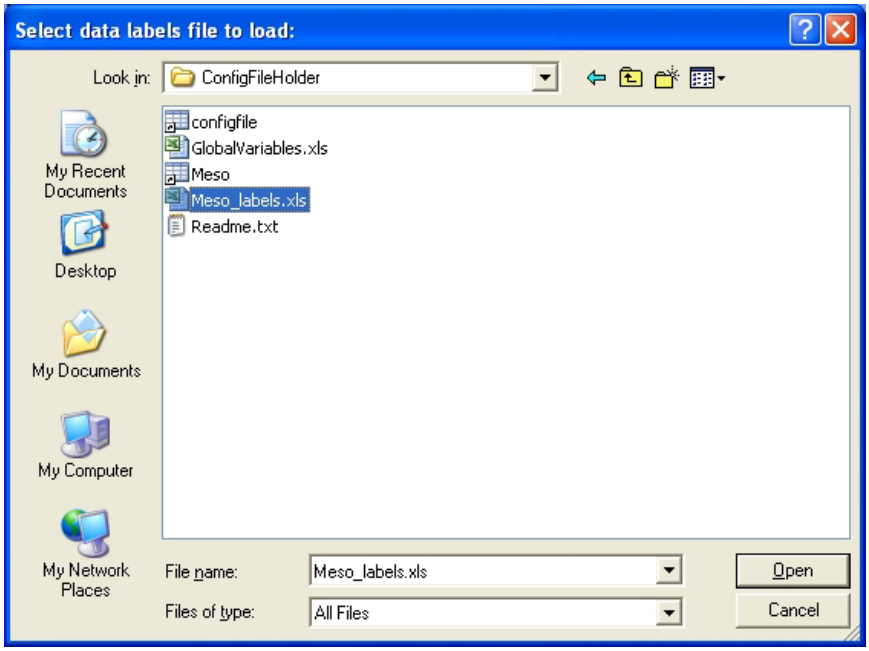

Figure 6.4. Data Labels Search dialog box.

Once the data labels is properly loaded, the name of the file is displayed in the Load Data Labels textbox and the Run button in the Preprocessing controls section is highlighted in red, signaling that data preprocessing is the next step in the application (see Figure 6.5).

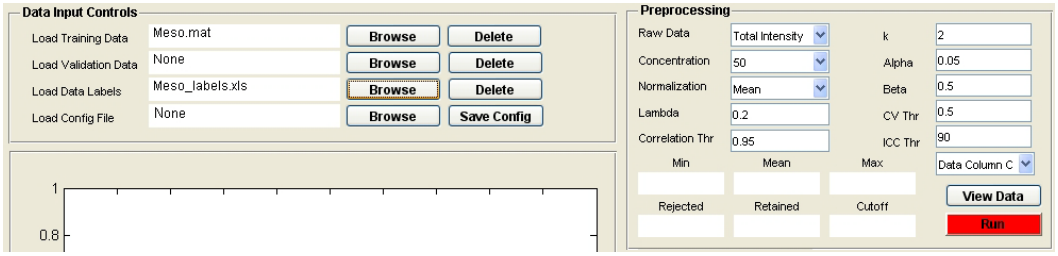

Figure 6.5. Data Input and Preprocessing Controls sections after the data labels have been loaded.

Check each of the preprocessing components in the Preprocessing Controls section to make sure each of the selected values is correct before conducting preprocessing. If any of the values are changed to values that are outside of acceptable limits, an error will be thrown, the incorrect value will be highlighted in orange, and the text from the error will be displayed in the Status/Error textbox in the Status and Error Controls section of the application. Run preprocessing by clicking the red Run button in the Preprocessing Controls section of the application. Once preprocessing is complete, the Min, Mean, Max, Rejected, and Retained

textboxes will be populated with values (the Cutoff textbox is not used at this time and is populated with TBD as a placeholder after preprocessing), the Run button in the Feature Selection/Projection Controls section will be highlighted in red, and the Control, Case, and Test checkboxes in the Feature Selection/Projection Controls section will become visible, displaying the name of each applicable cancer type in the study (see Figure 6.6).

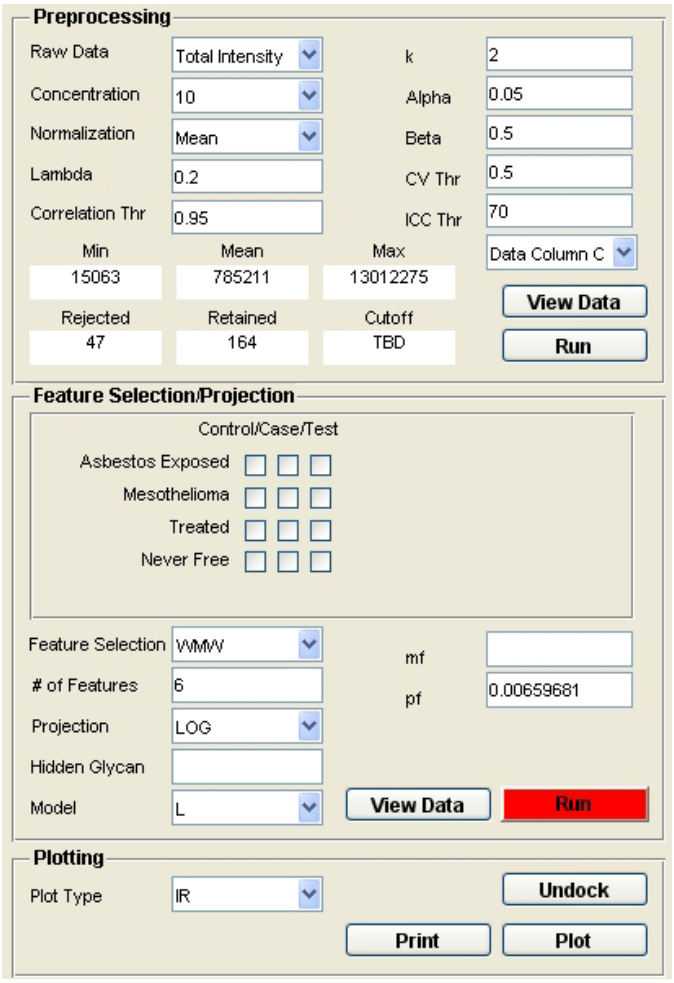

Figure 6.6. Preprocessing and Feature Selection/Projection Controls sections after preprocessing is completed.

The glycans that were rejected during preprocessing can be viewed in the Preprocessing window of the application. Clicking the View Data button in the Preprocessing Controls section opens the Preprocessing window (See Figure 6.7).

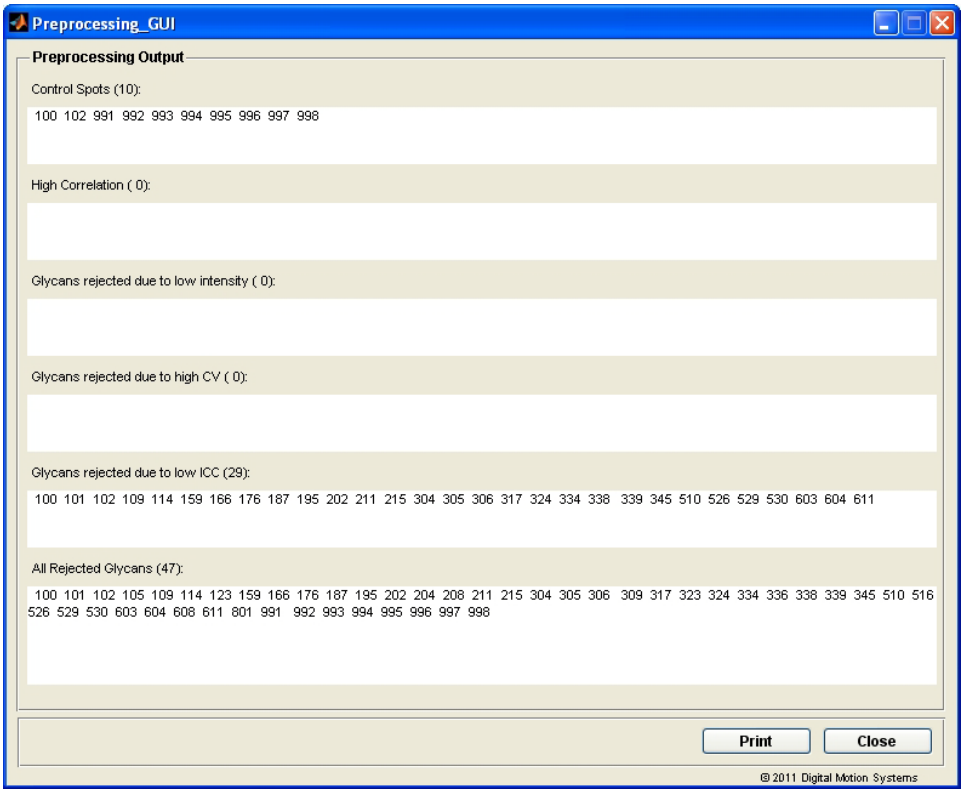

Figure 6.7. Preprocessing window after preprocessing is complete.

Check each of the feature selection and projection components in the Feature Selection/Projection Controls section and make sure each is correct before conducting the analysis of data. This includes checking appropriate checkboxes in the Control, Case, and Test columns. At least one checkbox representing a type of cancer must be checked in the Control and Case columns, but the same disease cannot be selected in both columns. If any of the checkboxes are selected in the Test column, the data is processed as validation data based on the class membership from training sets. For this example, Mesothelioma is selected as the Control group, Asbestos Exposed is selected as the Case group, and Treated is selected as the Test group (see Figure 6.8).

Run feature selection and projection by clicking the red Run button in the Feature Selection/Projection Controls section. Once the data analysis is complete, the values for mf and pf will be populated correctly and the Run button in the Plotting Controls section will be highlighted in red, signaling the next step in the application (see Figure 6.9).

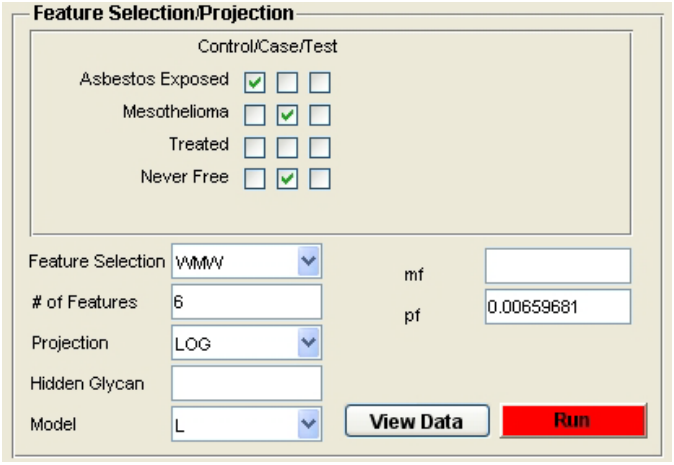

Figure 6.8. Checked checkboxes in the Feature Selection/Projection Controls section.

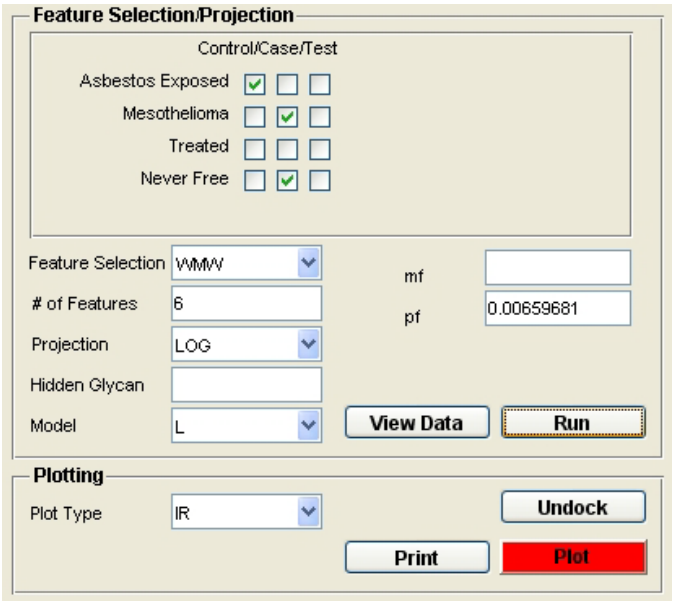

Figure 6.9. Feature Selection/Projection and Plotting Controls sections after feature selection and projection are completed.

The top features selected during data analysis and the order of those features can be viewed in the Output window by clicking the View Data button in the Feature Selection/Projection Controls section (see Figure 6.10).

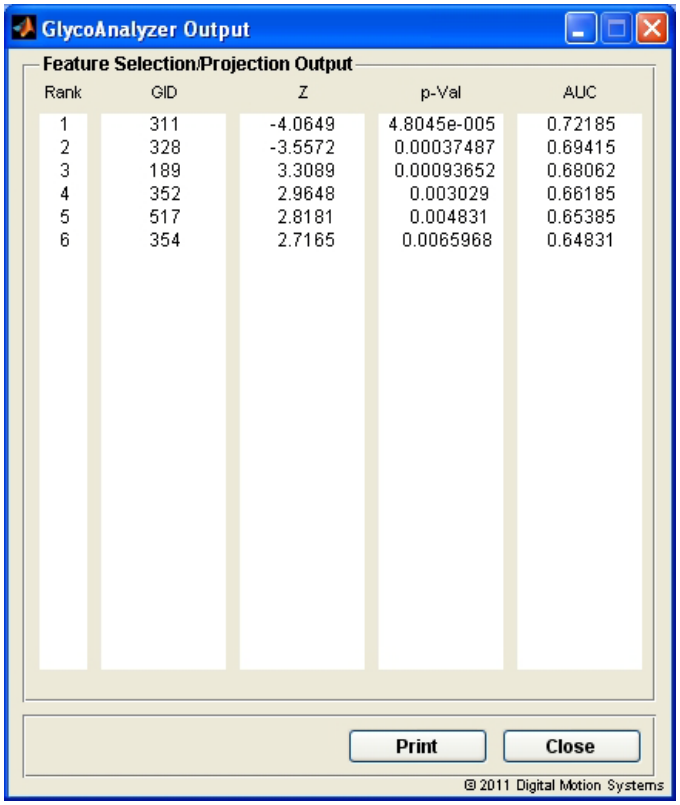

Figure 6.10. Output window after feature selection and projection are complete.

Before plotting the data, select the desired plot type from the Plot Type pop-up menu. The selected plot determines which controls are visible in the Plotting Controls section. For the first example, IR is selected from the Plot Type pop-up menu, signaling that an ImmunoRuler plot will be drawn. The visible plotting controls for an ImmunoRuler plot include the Threshold and Patient radio buttons, Sort pop-up menu, Decision Point pop-up menu, and Clear Tips button. Clicking the Run button in the Plotting Controls section of the application will plot the ImmunoRuler plot in the Main axis of the application. Once the plot is complete, the values for Sn, Sp, PPV, NPV, ACC, and AUC will be updated with correct values and the number of patients in each set will be listed in the graph legend (see Figure 6.11).

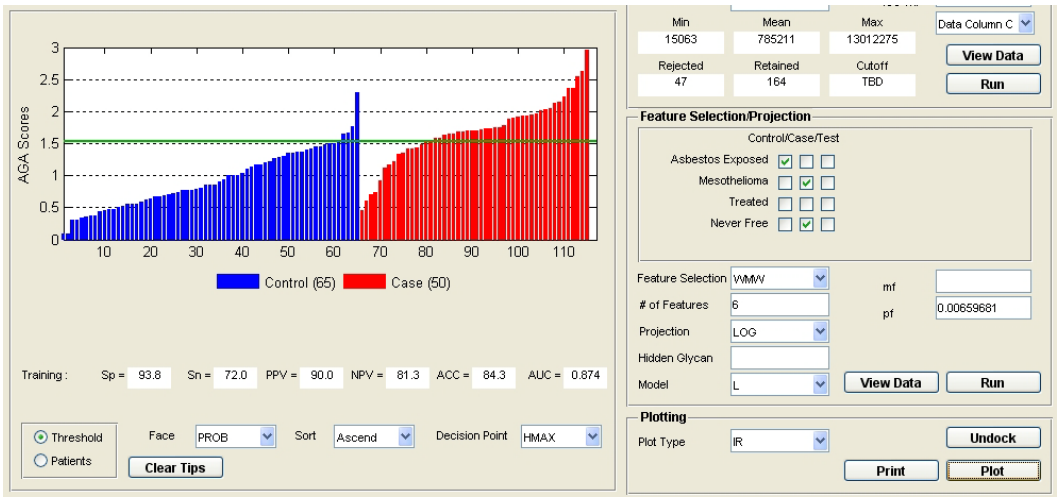

Figure 6.11. Completed ImmunoRuler plot.

Once the plot is complete, a larger view of the graph can be displayed in the Plot window of the application. Clicking the Undock button opens the Plot window (see Figure 6.12).

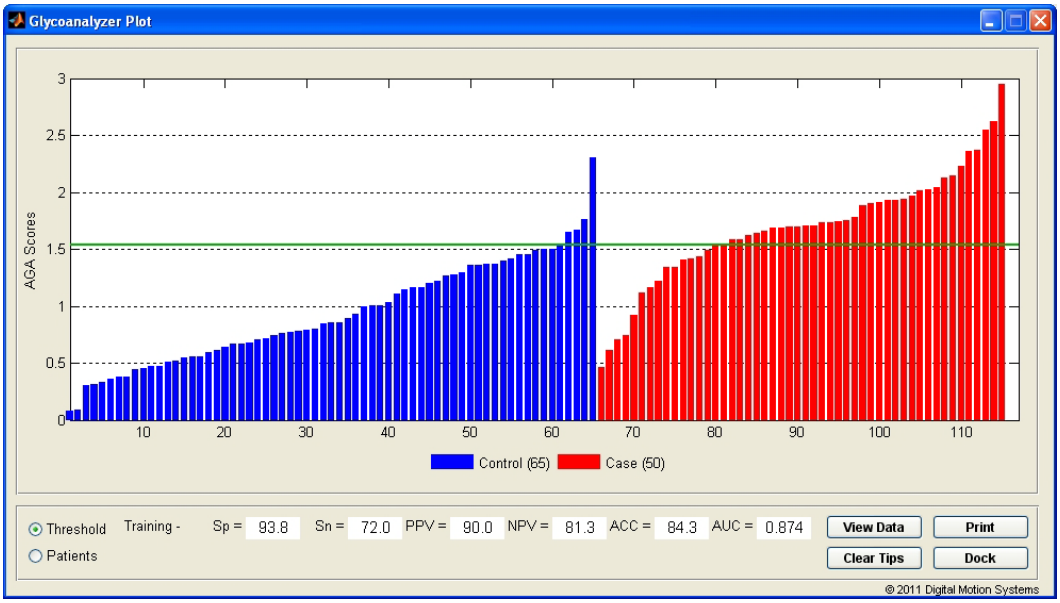

Figure 6.12. Plot window after completed ImmunoRuler plot.

In the main window, the plotted threshold line can be changed in the Main axis of the application. To do this, make sure the Threshold radio button is selected in the Plotting Controls section. Click on any of the white space on the axis above or below the threshold line to change the height. Once the threshold line is replotted, the values for Sn, Sp, PPV,

NPV, and ACC are updated to reflect the new height of the threshold line (see Figure 6.13). This feature works the same way for both the Main axis in the Main window and in the Plot window axis.

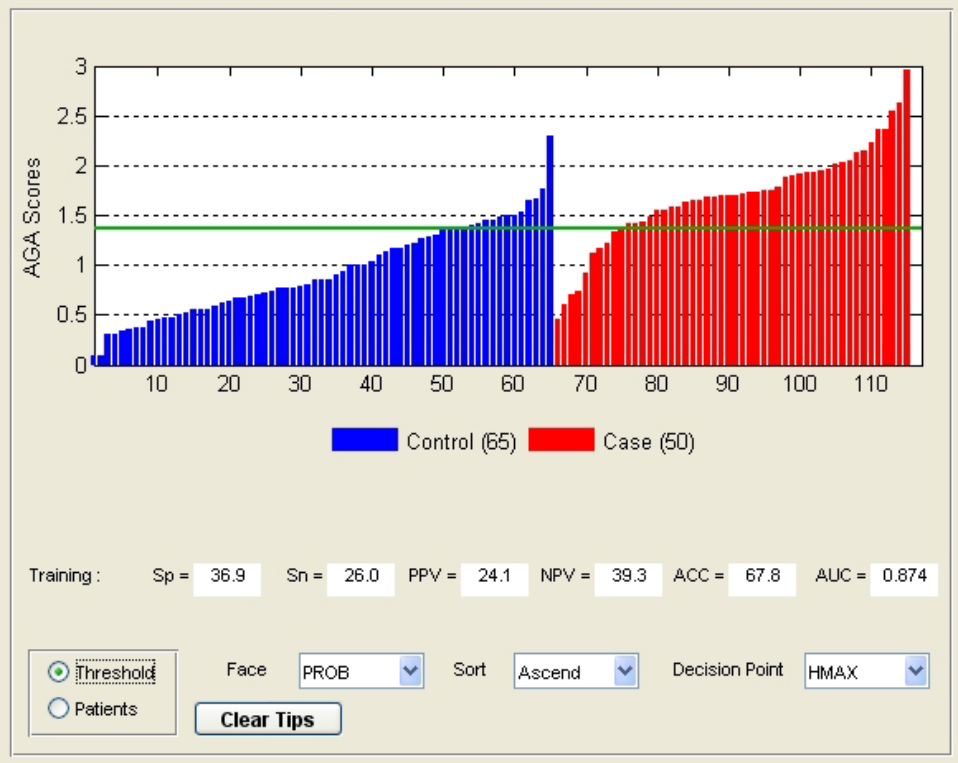

Figure 6.13. Replotted ImmunoRuler after a change in the threshold height.

Viewing intensity information about each patient in the study can be achieved by clicking the Patients radio button in the Plotting Controls section and then clicking on one of the colored ImmunoRuler bars. A tool tip appears detailing the patient's identification number and calculated intensity value. Clicking on a new patient erases the tool tip from the previous patient and creates a new tool tip with the new patient's details. Clicking on the Clear Tips button deletes a tool tip from the graph (see Figure 6.14). This feature works both in the Main axis in the Main window and in the Plot window axis.

Once data analysis is complete, the type of plot can be changed to view the data output in different ways. Updating the type of plot involves changing the value in the Plot Type pop-up menu. Selecting either the PDF or ROC plots deletes all of the controls for the ImmunoRuler plot. The only control for either type of plot is the pop-up menu that selects if

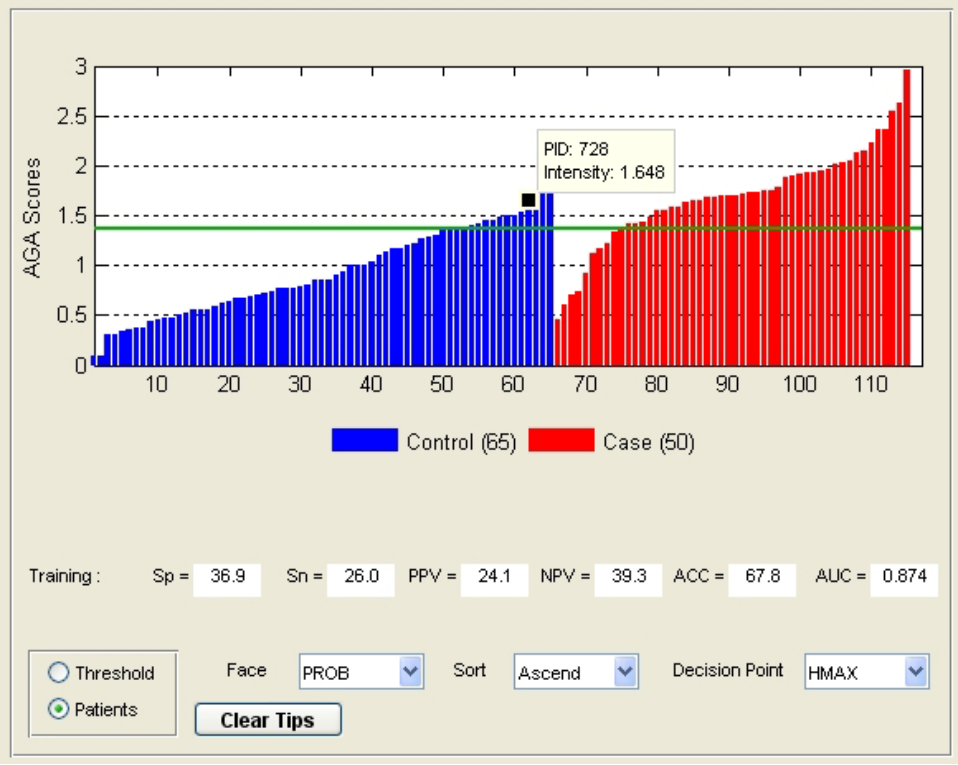

Figure 6.14. ImmunoRuler tool tip.

individual plots for each top feature (up to six) or a combined plot of all of the top features is plotted.

For the next example, a combined and individual ROC plots will be created and displayed. If any control is changed, the Plot button in the Plotting Controls section will be highlighted in red, signaling that the plot should be run again. Selecting INDIVIDUAL in the menu below the Plot Type and clicking the Plot button will create the individual ROC plots. In this case, six individual plots will be created because there are six top features specified in the Number of Features editable textbox in the Feature Selection/Projection Controls section of the application (see Figure 6.15). In each plot, the glycan number and AUC-value are displayed above each individual plot.

Plotting the combined ROC plot for all top six features is completed by changing the pop-up menu to COMBINED and clicking the Run button in the Plotting Controls section (see Figure 6.16). The top six glycan numbers and combined AUC-value are displayed above the plot in the header.

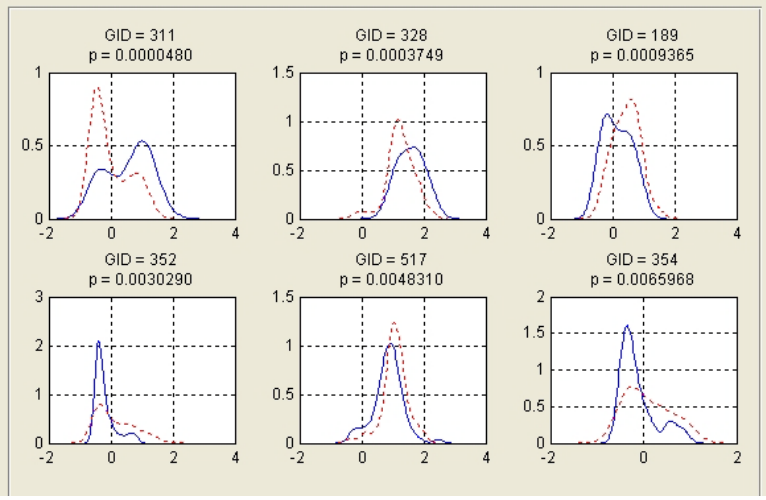

Figure 6.15. Individual ROC plots for six top features.

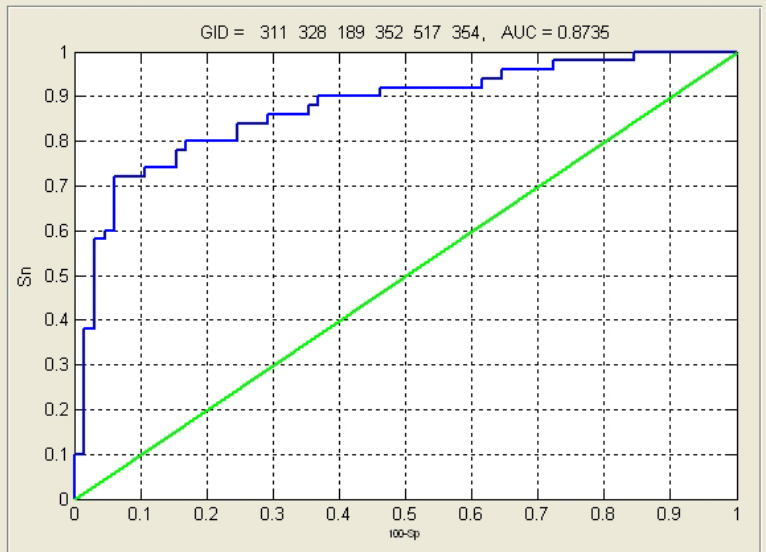

Figure 6.16. Combined ROC plot for six top features.

For the next example, a combined and individual PDF plots will be created and displayed. If any control is changed, the Plot button in the Plotting Controls section will be highlighted in red, signaling that the plot should be run again. Selecting Individual in the menu below the Plot Type and clicking the Plot button will create the individual PDF plots. In this case, six individual plots will be created because there are six top features specified in the Number of Features editable textbox in the Feature Selection/Projection Controls section of the application (see Figure 6.17). In each plot, the glycan number and p-value are displayed above each individual plot.

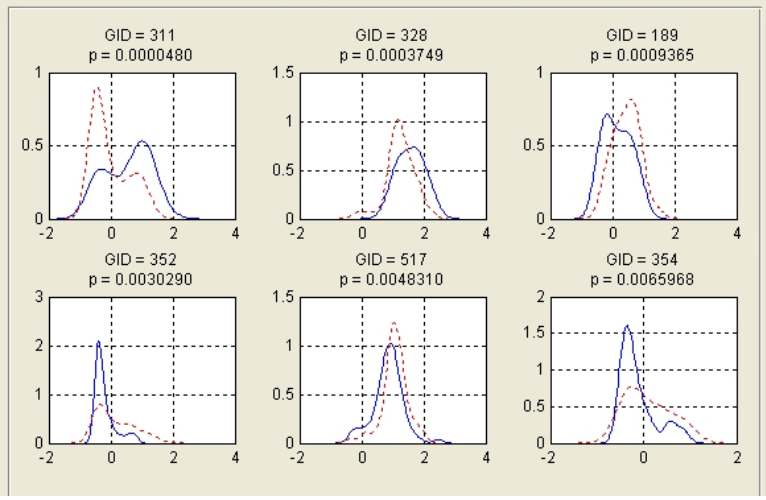

Figure 6.17. Individual PDF plot for six top features.

Plotting the combined ROC plot for all top six features is completed by changing the pop-up menu to COMBINED and clicking the Run button in the Plotting Controls section (see Figure 6.18). The top six glycan numbers and combined -value are displayed above the plot in the header.

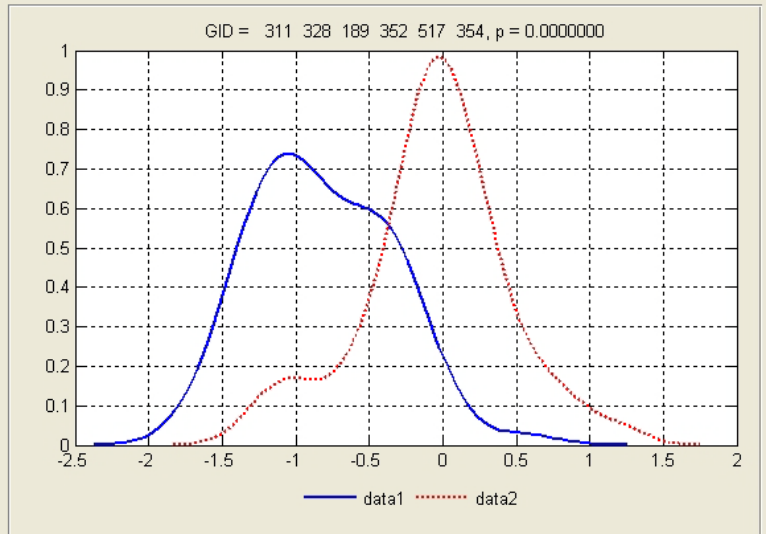

Figure 6.18. Combined PDF plot for six top features.

One the data has been plotted, the GlycoAnalyzer application can be reset by clicking the Reset button in the Status/Error Controls section. In order complete the reset, the user has to verify the reset in a Reset Question dialog box. The current configuration of all application components may also be saved by clicking the Save Config button in the Data

Input Controls section of the application and using the standard Windows save dialog box to create the name and browse for a location of the configuration file. This configuration can be reloaded at any time to bring the GlycoAnalyzer back to the same configuration. To close the application, click the Close button in the Status/Error Controls section of the application and verify the close in the Quit dialog box.

CHAPTER 7

MOBILE GLYCOANALYZER

The GlycoAnalyzer application is still in the early stage of development. Currently, the compiled application runs on a single workstation. All of the required libraries are available, via the MCRInstaller, and all data processing and plotting is done on that single workstation. The development, compilation, and packaging were completed entirely in the MATLAB development environment.

In the future, an idea is to make the GlycoAnalyzer into a networked, client-server solution. The client-side application would run on Android and iOS devices that communicate wirelessly with the server-side running the data processing engine. Patient data and data labels will be loaded into a basic front-end application installed on the mobile device. This application will contain the same components as the current GlycoAnalyzer application. Once the user has selected options for preprocessing, feature selection, projection, and plotting, the data, loaded initially, would be sent directly to the server for processing. As soon as processing is complete, the final information is sent back to the mobile device for display and plotting. The full version of MATLAB will be running on the server and will handle the bulk of the required data processing. While mobile devices are becoming more powerful each year, a client-server solution relieves the need for expensive, time consuming mobile processing. It also shortens the development of the entire solution, because many of the files required would not need to be ported from MATLAB to Objective-C or Java, neither of which have the built in libraries MATLAB has for scientific programming.

Currently, a basic, non-functional, front-end iOS application has been built using Objective-C and Cocoa for iPad to showcase the ability to create a client solution that models the current GlycoAnalyzer application components and workflow. Figures 7.1, 7.2, and 7.3 detail some of the screen mockups on this very early prototype. While this is still a nonfunctioning mock-up, it shows the potential of the GlycoAnalyzer for growth and future development on different platforms.

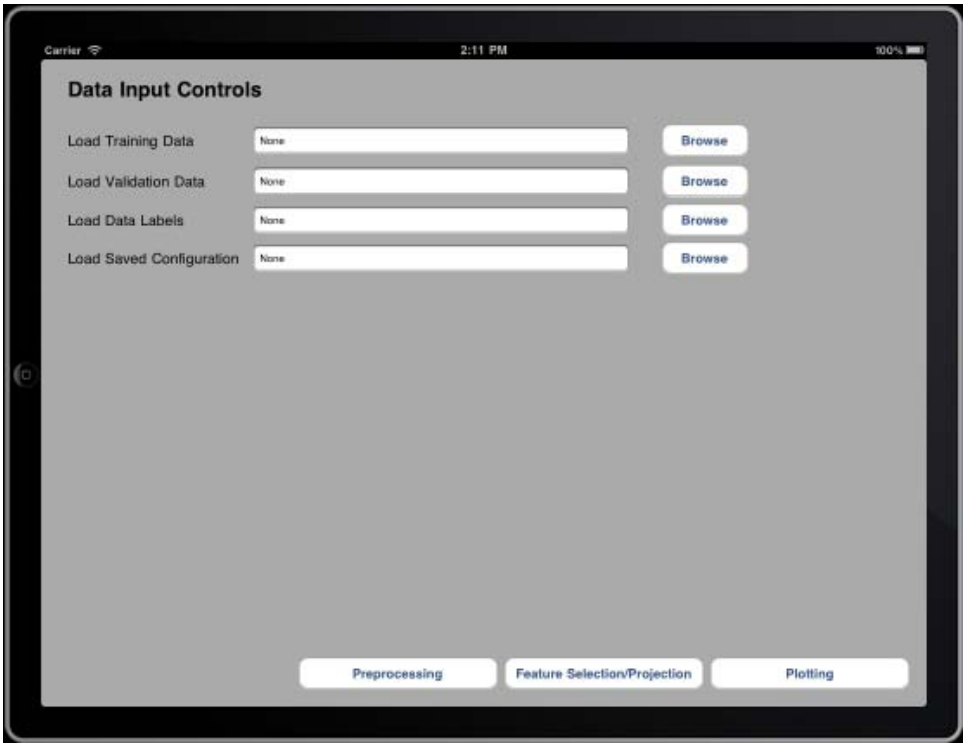

Figure 7.1. Data Input Controls running on iOS.

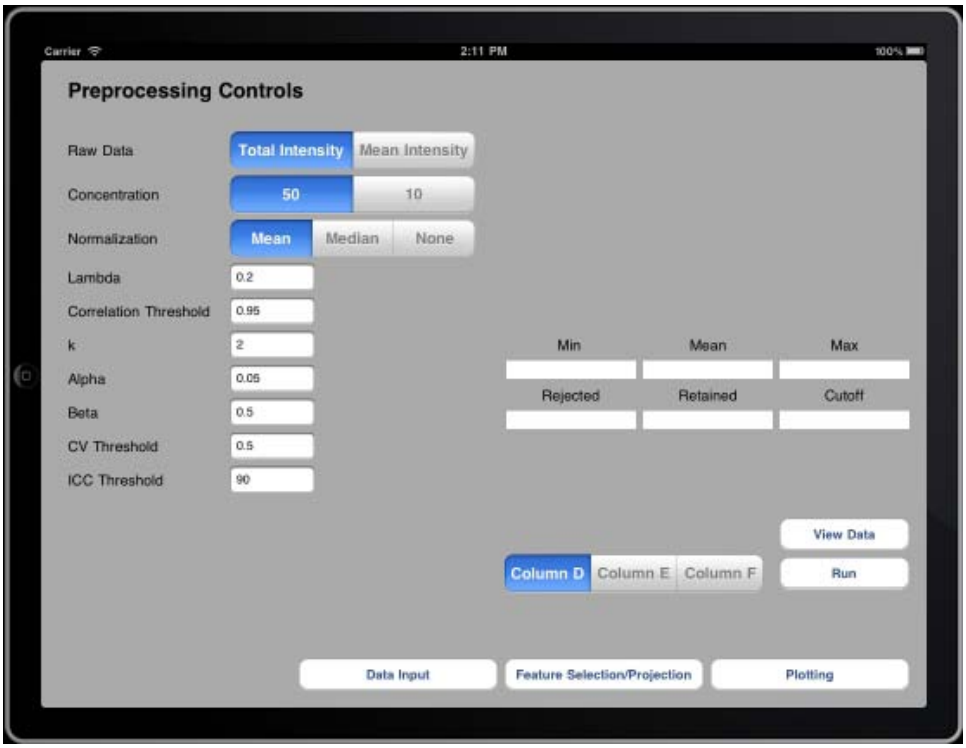

Figure 7.2. Preprocessing Controls running on iOS.

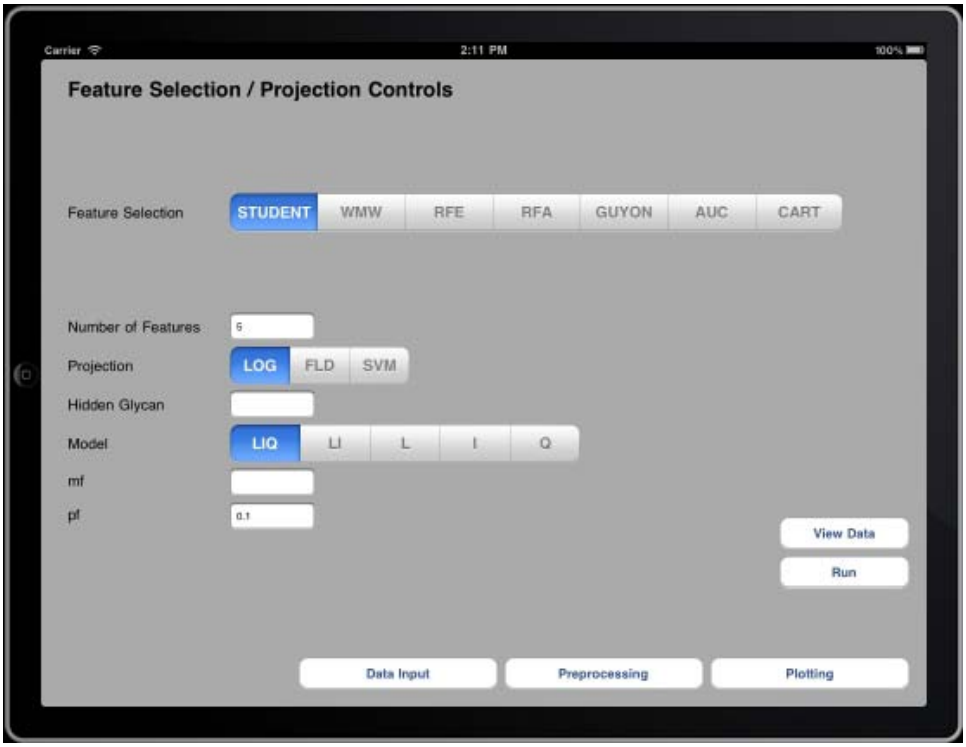

Figure 7.3. Feature Selection and Projection Controls running on iOS.

CHAPTER 8

CONCLUSION

This paper specified the functionality and concepts behind the creation of the GlycoAnalyzer, detailed the implementation of the application in the MATLAB environment, and discussed the compilation, packaging, and installation of the standalone executable application used on end-user workstations. The document also includes comprehensive demonstration of all aspects of the application listed above, including a short version of end user work flow.

The GlycoAnalyzer application represents the first step in taking the many data analysis functions and successfully integrating them into a fully functioning graphical user interface. The complex interaction between the application support functions and the data analysis engine has evolved over time as the library of data analysis functions has changed and become more complex.

Throughout the process of designing the application, there were many design changes, making the full application more functional, modular, and user friendly. Some of these changes involved layout changes that added functionality and additional features, including adding a hidden glycan feature, adding additional ways of plotting data, and adding extra windows that display additional information in the Preprocessing and Feature Selection and Projection Controls sections. Some of the changes make updating the application easier, such as creating functions that work within and outside of the application so that each time the library of data analysis functions are updated, they can be copied to the correct directory and immediately work with the GlycoAnalyzer. Finally, some of the changes involve making the application easier to use for developers and end-users, including the creation of additional error checking, more detailed error text, and a way to find the exact function and line of code where an error is thrown so that the end-user can detail exactly what he is seeing when there is a run-time error. This last feature makes finding and fixing errors easier for the development team. Work is still being completed on increasing display output control on the

data analysis engine functions so that the final compiled application will run smoothly on end-user workstations.

Future work on the GlycoAnalyzer will increase usability while incorporating new functionality, including classifier evaluation, such as cross validation and bootstrapping, adding additional feature selection methods, such as random forest and ant colony algorithms, and adding new ways of graphing data, such as scatterplots and boxplots. Finally, the development of the mobile application discussed in Chapter 7 seems to be an attractive solution that will allow users to run the program anywhere there is an internet connection.

REFERENCES

[1] AMERICAN CANCER SOCIETY, *American Cancer Society guidelines for the early detection of cancer*. American Cancer Society, http://www.cancer.org/healthy/findcancerearly/cancerscreeningguidelines/american-cancer-society-guidelines-for-the-early-detection-of-cancer, accessed June 2011, 2010.

[2] T. W. HUTCHENS AND Y-T YIP, *New desorption strategies for mass spectrometric analysis of macromolecules*, Rapid Comm. Mass Spectrometry, 7 (1993), pp. 576-580.

[3] G. L. WRIGHT JR., *SELDI proteinchip MS: A platform for biomarker discovery and cancer diagnosis*, Expert Rev. Mol. Diag., 2 (2002), pp. 549-563.

[4] H. J. ISSAQ, T. D. VEENSTRA, T. P. CONRADS, AND D. FELSCHOW. *The SELDI-TOF MS approach to proteomics: Protein profiling and biomarker identification*, Biochem. Biophys. Res. Comm., 292 (2002), pp. 587-592.

[5] D. SIDRANSKI, *Nucleic acid-based methods for detection of cancer*, Sci., 278 (1997), pp. 1054-1058.

[6] P. O. BROWN AND D. BOTSTEIN, *Exploring the new world of genome with DNA microarrays*, Nat. Gen., 21 (1999), pp. 33-37.

[7] M. I.VUSKOVIC, H. XU, N. V. BOVIN, H. I. PASS, AND M. E. HUFLEJT, *Processing and analysis of printed glycan array data for early detection, diagnosis, and prognosis of cancers*. Unpublished report, 2011.

[8] N. V. BOVIN AND M. E. HUFLEJT. *Unlimited glycochip*, Trends Glycosci. Glycotechnol., 20 (2008), pp. 245-258.

[9] M. E. HUFLEJT, M. VUSKOVIC, D. VASILIU, H. XU, P. OBUKHOVA, N. SHILOVA, A. TUZIKOV, O. GALANINA, B. ARUN, K. LU, AND N. BOVIN, *Anti-carbohydrate antibodies of normal sera: Findins, surprises, and chanllenges*, Mol. Immunol., 46 (2009), pp. 3037-3049.

[10] L. I-K. LIN, *A concordance correlation coefficient to evaluate reproducibility*, Biomet., 45 (1989), pp. 255-268.

[11] MEDCALC, *Concordance correlation coefficient*. MedCalc, http://www.medcalc.org/manual/concordance.php, accessed October 2011, 2011.

[12] H. X. BARNHART, M. HABER, AND J. SONG, *Overall concordance correlation coefficient for evaluating agreement among multiple observers*, Biomet., 58 (2002), pp. 1020–1027.

[13] J. A. JOHN AND N. R. DRAPER, *An alternative family of transformations*, Appl. Stat., 29 (1980), pp. 190-197.

[14] D. R. CAPRETTE, *Student's t test (for independent samples)*. Experimental Biosciences, http://www.ruf.rice.edu/~bioslabs/tools/stats/ttest.html, accessed March 2011, 2005.

[15] W. M. K. TROCHIM, *The t-test*. Social Research Methods, http://www.socialresearchmethods.net/kb/stat_t.php, accessed March 2011, 2006.

[16] C. WILDE AND G. SEBER, *The Wilcoxon rank-sum test*. University of Auckland, http://www.stat.auckland.ac.nz/~wild/ChanceEnc/Ch10.wilcoxon.pdf, accessed March 2011, n.d.

[17] R. L. OTT AND M. T. LONGNECKER, *An Introduction to Statistical Methods and Data Analysis*, Cengage Learning, Belmont, California, 2010.

[18] D. K. NEAL, *The rank sum test*. Western Kentucky University, http://www.wku.edu/~david.neal/statistics/nonparametric/ranksum.html, accessed September 2011, 2003.

[19] V. N. VAPNIK, *The Nature of Statistical Learning Theory*, Springer, New York, 1995.

[20] M. BROWN, *Support vector machines*. University of California, Santa Cruz, http://compbio.soe.ucsc.edu/genex/genexTR2html/node9.html, accessed October 2011, 2005.

[21] C. E. METZ, *Basic principles of ROC analysis*, Nuc. Med.Sem., VIII (1978) pp. 283-298.

[22] MEDCALC, *ROC curve analysis: Introduction*. MedCalc, http://www.medcalc.be/manual/roc.php, accessed November 2009, 2009.

[23] D. HAND AND R. TILL, *A simple generalization of the area under the ROC curve for multiple class classification problem*, Mach. Learn., 45 (2001), pp. 171-186.

[24] T. FAWSETT, ed., *ROC graphs: Notes and practical considerations for researchers*, in Technical Report, HPL-2003-4, Intelligent Enterprise Technologies Laboratory, HP Laboratories, Palo Alto, California, 2003.

[25] P. FLACH, ed., *Proceedings of the 21st International Conference on Machine Learning*, Banff, Canada, 2004, ICML.

[26] J. M. HANLEY AND B. J. MCNEIL, *The meaning of use of the area under a receiver operating characteristic (ROC) curve*, Radiol., 143 (1982), pp. 29-36.

[27] A. P. BRADLEY, *The use of the area under the roc curve in the evaluation of machine learning algorithm*, Patt. Rec., 30 (1997), pp. 1145-1159.

[28] C. D. MANNING, P. RAGHAVAN, AND H. SCHÜTZE, *A Guide to Information and Retrieval*, Cambridge University Press, Cambridge, England, 2009.

[29] MATHWORKS, *Ttest2*. Mathworks, http://www.mathworks.com/help/toolbox/stats/ttest2.html, accessed October 2011, n.d.

[30] MATHWORKS, *Ranksum*. Mathworks, http://www.mathworks.com/help/toolbox/stats/ranksum.html, accessed October 2011, n.d.

[31] I. GUYON AND A. ELISSEEFF, *An introduction to variable and feature selection*, J. Mach. Learn. Res., 3 (2003), pp. 1157-1182.

[32] M. BROWN, *Fisher's linear discriminate*. University of California, Santa Cruz, http://compbio.soe.ucsc.edu/genex/genexTR2html/node12.html, accessed October 2011, 2005.

[33] M. I. VUSKOVIC AND M. E. HUFLEJT, *System, method and computer-accessible medium for evaluating a malignancy status in at- risk populations and during patient treatment management*, Patent 61/318,144, Dorsey and Whitney LLP No. P215746.US.01 – 475396-00261, March 2010.

[34] N. M. ADAMS AND D. J. HAND, *Comparing classifiers when the misclassification costs are uncertain*, Patt. Rec., 32 (1999), pp. 1139-1147.

[35] MATHWORKS, *Ksdensity*. Mathworks, http://www.mathworks.com/help/toolbox/stats/ksdensity.html, accessed September 2011, n.d.

[36] MATHWORKS, Laying out a GUI. Mathworks, http://www.mathworks.com/help/techdoc/learn_matlab/f5-999222.html, accessed October 2011, n.d.

[37] MATHWORKS, *Guide*. Mathworks, http://www.mathworks.com/help/techdoc/ref/guide.html, accessed October 2011, n.d.

[38] MATHWORKS, *Files generated by GUIDE*. Mathworks, http://www.mathworks.com/help/techdoc/creating_guis/f10-1005070.html, accessed October 2011, n.d.

[39] MATHWORKS, *Function_handle (@)*. Mathworks, http://www.mathworks.com/help/techdoc/ref/function_handle.html, accessed October 2011, n.d.

[40] MATHWORKS, *Handle graphics and properties guide*. Mathworks, http://www.mathworks.com/support/tech-notes/1200/1205.html, accessed October 2011, n.d.

[41] MATHWORKS, *Align components*. Mathworks, http://www.mathworks.com/help/techdoc/creating_guis/f8-998370.html, accessed October 2011, n.d.

[42] MATHWORKS, *Set*. Mathworks, http://www.mathworks.com/help/techdoc/ref/set.html, accessed October 2011, n.d.

[43] MATHWORKS, *Standalone applications introduction*. Mathworks, http://www.mathworks.com/help/toolbox/compiler/f7-963587.html, accessed September 2011, n.d.

[44] MATHWORKS, *Standalone executable*. Mathworks, <http://www.mathworks.com/help/toolbox/compiler/f10-999433.html, accessed September 2011, n.d.

[45] MATHWORKS, *Working with the MCR*. Mathworks, http://www.mathworks.com/help/toolbox/compiler/f12-999353.html, accessed September 2011, n.d.

[46] MATHWORKS, *Supported and compatible compilers – Release 2010a*. Mathworks, http://www.mathworks.com/support/compilers/R2010a/win32.html, accessed September 2011, n.d.

[47] MATHWORKS, *Deploytool*. Mathworks, http://www.mathworks.com/help/toolbox/compiler/deploytool.html, accessed September 2011, n.d.

[48] MATHWORKS, *Magic square example: Creating a standalone executable or shared library from MATLAB code*. Mathworks, http://www.mathworks.com/help/toolbox/compiler/bsl9c8_.html, accessed September 2011, n.d.

APPENDIX A

GLYCOANALYZER COMPONENT DESCRIPTIONS

This section details the functionality of each button, pop-up menu, editable textbox, static textbox, checkbox, radio button, and axis included in the GlycoAnalyzer application. The information in this appendix makes up the main information found in the GlycoAnalyzer help file which can be accessed by pressing the Help button in the Status and Error Section of the application.

Data Input Controls Section:

Push Buttons:

Browse for Training Data: Clicking the Browse button opens a Windows Search dialog box allowing the user to select a MAT-file that contains training data. If the data file is in the correct format, it will be loaded as soon as the user clicks the Open button in the dialog box. If the file is not correct for any reason, an error will be thrown and the user will be directed to open a correct file. Once a data file is loaded, the filename will be displayed in the static textbox to the left of the Browse button.

Delete Training Data: Clicking the Delete button opens a dialog box allowing the user to verify that the training data file will be deleted. Clicking the Yes button in the dialog box deletes the file and all of the data from the GlycoAnalyzer. Once the training data has been deleted, the static textbox to the left of the Delete button will display the word, "None." Clicking the No button in the dialog box will retain the training data in the application and close the dialog box with no change to the application.

Browse for Validation Data: Clicking the Browse button opens a Windows Search dialog box allowing the user to select a MAT-file that contains validation data. If the data file is in the correct format, it will be loaded as soon as the user clicks the Open button in the dialog box. If the file is not correct for any reason, an error will be thrown and the user will be directed to open a correct file. Once a data file is loaded, the filename will be displayed in the static textbox to the left of the Browse button.

Delete Validation Data: Clicking the Delete button opens a dialog box allowing the user to verify that the training data file will be deleted. Clicking the Yes button in the dialog box deletes the file and all of the data from the GlycoAnalyzer. Once the validation data has been deleted, the static textbox to the left of the Delete button will display the word, "None." Clicking the No button in the dialog box will retain the validation data in the application and close the dialog box with no change to the application.

Browse for Data Labels: Clicking the Browse button opens a Windows Search dialog box allowing the user to select a XLS-file that contains data labels that go with the loaded training data. If the data labels file is in the correct format, it will be loaded as soon as the user clicks the Open button. If the file is not correct for any reason, an error will be thrown and the user will be directed to open a correct file. Once a data labels file is loaded, the filename will be displayed in the static textbox to the left of the Browse button.

Delete Data Labels: Clicking the Delete button opens a dialog box allowing the user to verify that the data labels file will be deleted. Clicking the Yes button in the dialog box deletes the file and all of the labels from the GlycoAnalyzer. Once the training date has been deleted, the static textbox to the left of the Delete button will display the word, "None." Clicking the No button in the dialog box will retain the data labels in the application and close the dialog box with no change to the application.

Browse for the Configuration File: Clicking the Browse button opens a Windows Search dialog box allowing the user to select a MAT-file that contains configuration information for the GlycoAnalyzer. If the configuration file is in the correct format, it will be loaded as soon as the user clicks the Open button in the dialog box. Automatically, all of the application components will immediately be set to the configuration specified by the loaded configuration file. If the file is not correct, an error will be thrown and the user will be directed to open a correct file.

Once the configuration file is loaded, the filename will be displayed in the static textbox to the left of the Browse button.

Save Config: Clicking the Browse button opens a Windows dialog box allowing the user to save the entire application configuration as a MAT-file. Once the configuration file is saved, it can be loaded back into the application by browsing for the file.

Preprocessing Controls Section:

Pop-up Menus:

Raw Data: The Raw Data pop-up menu allows the user to select between Total Intensity and Raw Intensity. The value Total Intensity of summarized glycan spots represents raw data read from the slide and represents a measure of the binding level of AGA. The value Mean Intensity of summarized glycan spots represents preprocessed, averaged data that has been read from different batches of slides during different days. The data is averaged using median because the readings are more accurate than if the mean was used.

Concentration: The PGA used during these tests contains glycans that are attached to the slides in two different concentrations for both florescence intensities; 10 and 50 μM. The Concentration pop-up menu allows the user to select either of these concentrations of glycans during the preprocessing phase.

Normalization: The Normalization pop-up menu represents the normalization style used during the normalization phase of preprocessing. The three options are: MEAN, MEDIAN, and NONE, where if NONE is selected, no normalization takes place.

Editable Textboxes:

k: The value, k, screens all features and removes a feature if all but k patients are above the threshold, s_α. The value k must be an integer between zero and a fraction of the number of patients in the training set. The higher the value of k, the more glycans will be rejected. If k=0, the feature is

rejected if all of the features are at a level less than the threshold. If k=1, the feature is rejected if at least two features are above the threshold.

Alpha (α): The value, α, is a noise screening parameter in the threshold, s_α. The value, α must be greater than 0.001 and less than 0.99. This value is used in conjunction with the parameter k to screen out all glycans with intensities that are at, or below, the value, s_α, for at least n-k patients.

Beta (β): The variable, β, is used in conjunction with the CV Threshold and represents a percentage of patients. This value must be greater than 0.05 and less than 0.95. If β is 0.6, then all glycans would be rejected if 60% of the patients were at or above the CV Threshold percentage of the coefficient of variation.

CV Thresh: The CV Threshold is used in conjunction with the variable, β, and is a percentage of the coefficient of variation. The value must be greater than 0.00001 and less than 0.99. This value is used to screen out all features where a percentage, β, of the patients are at or above the CV Threshold percentage of the coefficient of variation.

Non-Editable Textboxes:

Min: Minimum raw fluorescence intensity in the matrix D.X.

Mean: Mean raw fluorescence intensity of all values in the matrix D.X.

Max: Maximum raw fluorescence intensity in the matrix D.X.

Rejected: Number of glycans rejected during preprocessing.

Retained: Number of retained glycans after preprocessing.

Cutoff: Not used in this version of the GlycoAnalyzer. This textbox will be used in a future version of the application.

Push Buttons:

View Data: The View Data button opens the Preprocessing window. If the preprocessing of data is not complete, the Preprocessing window opens and all of the non-editable textboxes are blank. Once preprocessing is complete, the rejected glycan numbers are populated in the non-editable textboxes.

Run: Clicking the Run button in the Preprocessing Controls section starts the preprocessing of data. The Run button can only be clicked after the training data and labels have been successfully loaded and the button is colored red. Clicking the button at any other time throws an error and directs the user to the error condition.

Feature Selection and Projection Controls Section:

Pop-up Menus:

Feature Selection: The Feature Selection pop-up menu allows the user to select the desired feature selection type used in processing the data. The current choices are; WMW, Student, RFA, RFA_L, RFE, FFA, GUYON, AUC, MWA, RFA-CV, and CART.

Projection: The Projection pop-up menu allows the user to select the desired projection type used in processing the data. The current choices are; LOG, FLD, and SVM.

Modal: The Modal pop-up menu lists a feature that has not yet been implemented. Currently, the only available Modal value is 'L'. This feature will be implemented in future version of the GlycoAnalyzer application.

Editable Textboxes:

Number of Features: The Number of Features editable textbox represents the number of features which are used to combine the corresponding intensities into a single scalar value. If m=5, the application will consider 5 features. The number entered must be a positive integer between 1 and the number of total features in the assay library used in the study. This value does not include the hidden glycan, which, when not one of the calculated top ranked features, is included in the list of top ranked features.

Hidden Glycan: The hidden glycan is included during the feature selection and projection process. Even if the feature is not one of the selected features that remain after prefiltering, the hidden glycan is automatically included in the group of top features. The hidden glycan must be a

glycan in the original set of glycans. If the glycan listed in the Hidden Glycan editable text box is not a glycan in the original set of glycans an error is thrown and the user is directed to enter a correct glycan number.

mf: mf is used as a prefiltering value during the feature selection process. mf represents the number of Wilcoxon-ranked features that will be used in the feature selection process. While mf can be left blank by the user, if entered, it must be a positive integer between the number listed in the Number of Features textbox and the number of total features in the study.

pf: pf is the cutoff probability used to determine the number of candidate features. The candidate features are the top Wilcoxon-ranked features which have a p-value less than or equal to pf. Pf is an alternative to mp for defining prefiltering.

Check Boxes: The checkboxes allow users to select the Control, Case, and Test classes from a list of available disease classifications available in the training dataset. The Control column refers to patients who do not have the specified diseases while the Case column refers to patients who do have the specified diseases. The Test column is used to test the same dataset against the findings from the Control and Case classes. Users may check as many checkboxes in the Control and Case columns, but must select at least one of checkbox from each column. Errors are thrown if the user checks both the Control and Case checkboxes for the same disease or if no checkbox in either column. Checkboxes in the Test column may be the same as those checked in either the Control or Case columns. If a validation dataset is loaded into the GUI in the Data Input Controls section, the Test checkboxes disappear and are not checkable. The checkbox labels are populated using the variable, LID. The GlycoAnalyzer application interface can support up to ten disease classifications.

Push Buttons:

View Data: Clicking the View Data button in the Feature Selection and Projection Controls section opens the Output window and displays

information about specific features to the user. Clicking this button only displays data once Feature Selection and Projection have been completed. Clicking the button before Feature Selection and Projection are complete displays an empty window with no labels or information.

Run: Clicking the Run button in the Feature Selection and Projection Controls section starts the feature selection and projection of data. The Run button can only be clicked after the preprocessing has been successfully completed. Clicking the button before preprocessing is complete throws an error and directs the user to the error condition.

Plotting Section:

Pop-up Menus:

Plot Type: The Plot Type pop-up menu allows the user to select different ways of plotting data. The choices are two ImmunoRuler plots (i.e. IR, IR New), a PDF plot, and a ROC plot. To select a new plot, change the value in the Plot Type pop-up menu and click the Plot button.

Sort: The Sort pop-up menu allows the user to sort the patient identifiers (PID) by intensity in either of the ImmunoRuler plots. The three choices are; ascending, descending, or none. To change the PID sorting, change the Sort pop-up menu value and click the Plot button. The Sort pop-up menu is only visible if the plot is either of the ImmunoRuler plots. If the selected plot type is PDF or ROC, the Sort pop-up menu becomes invisible and cannot be clicked by the user.

Plot Flag: The Plot Flag pop-up menu allows the user to select if the top features are plotted in a combined individual plot or in several individual plots. Up to six top features can be plotted at any time. The Type pop-up menu is only visible if the plot is either a PDF or a ROC plot. If the selected plot type is either of the ImmunoRuler plots, the Type pop-up menu becomes invisible and cannot be clicked by the user.

Decision Point: The Decision Point pop-up menu determines the decision point strategy used in finding class membership of the two clusters of data. The pop-up menu contains the four values: HMAX, MEAN,

MEDIAN, and COST. HMAX selects a corrected decision point determined by the maximal training hit rate. MEAN determines a corrected decision point based on the middle of the two cluster means. MEDIAN determines a corrected decision point based on the middle of the two cluster medians. Selecting COST causes the two cost editable textboxes to appear and allows the user to specify a corrected decision point based on the ratio of cost-of-FPR and cost-of-FNR. The Decision Point pop-up menu is only visible if the plot is either of the ImmunoRuler plots. If the selected plot type is PDF or ROC, the Decision Point pop-up menu becomes invisible and cannot be clicked by the user.

Face: The Face pop-up menu specifies how the risk scores, the cutoff value for the risk scores, and the cutoff for risk scores which corresponds to cost = '1/1' are calculated for the first of the two ImmunoRuler plots. The pop-up menu contains the three values: PROB, LOGODDS, and ODDS. The Phase pop-up menu is only visible if the plot is the first of the two ImmunoRuler plots. If the selected plot type is the second ImmunoRuler plot, PDF or ROC, the Phase pop-up menu becomes invisible and cannot be clicked by the user.

Editable Textboxes:

Cost: The cost editable textboxes appear when the user selects COST in the Decision Point pop-up menu. Cost is a ration of the cost-of-FPR and cost-of-FNR. The first checkbox is the cost-of-FPR and the second checkbox is the cost-of-FNR. The values entered in each checkbox must be integers between 1 and 100. The default value for both editable textboxes is 1.

Radio Buttons:

Threshold: Clicking the Threshold radio button allows the user to change the height of the threshold line displayed in the ImmunoRuler plot. Changing the height is achieved by clicking in the plot over or under the threshold line. When the height is changed, the values in the Training

and Validation static textboxes are updated accordingly. The Threshold radio button is only visible if the plot is either of the ImmunoRuler plots. If the selected plot type is PDF or ROC, the Threshold radio button becomes invisible and cannot be clicked by the user.

Patients: Clicking the Patients radio button allows the user to get information about patients in the ImmunoRuler plot. Clicking on any of the bars in the ImmunoRuler plot displays a tool tip that details the patient identifier (PID) and the intensity. The Patients radio button is only visible if the plot is either of the ImmunoRuler plots. If the selected plot type is PDF or ROC, the Patients radio button becomes invisible and cannot be clicked by the user.

Push Buttons:

Print: Clicking the Print button allows the user to print the graphical output of data to any networked printer. Initially, a Print Preview window appears allowing the user to adjust the image to fit the desired page layout. Pressing the Print button in the Print Preview window sends the image to the selected printer.

Undock: Clicking the Undock button opens the Plot window, displaying the plotted data in a larger window for the user. The plotted information displayed in the Plotting section is identical to the information displayed in the Plot window.

Plot: Clicking the Plot button plots the data using the desired type of plot, determined by the Plot Type pop-up menu. If the selected plot type is either of the ImmunoRuler plots, the data is plotted in a single axis. If the PDF or ROC plots are selected, the number of displayed axes is determined by the Type pop-up menu in the Plotting section and the Number of Features editable textbox in the Feature Selection and Projection Controls section.

Clear Tips: Clicking the Clear Tips button clears any patient information tool tips displayed in the plot. If no patient information tool tips are displayed, the Clear Tips button is disabled and has no functionality.

The Clear Tips button is only visible if the plot is either of the ImmunoRuler plots. If the selected plot type is PDF or ROC, the Clear Tips button becomes invisible and cannot be clicked by the user.

Axes: Clicking on any axes in the application opens the plotted information in a new window. The information is displayed in a larger axis that is easier to view and separate from the original plotted display, but has reduced functionality (i.e. the threshold line cannot be moved and the individual patient information is inaccessible).

Status and Error Controls Section:

Push Buttons:

Reset: Clicking the Reset button resets each component in the application to the initial condition. If the GlycoAnalyzer is closed immediately after a complete reset, this initial condition is saved to the configuration file and loaded the next time the application is launched by the user.

Help: Clicking the Help button displays a text file detailing the functionality of each of the components contained in the application. A brief detail of the functionality of each component can also be accessed via tooltip by hovering over each component for several seconds.

Close: Clicking the Close button saves the current configuration of the GlycoAnalyzer to the configuration file and closes all open application windows. The current configuration is immediately available the next time the GlycoAnalyzer is launched by the user. A dialog box appears allows the user to confirm that closing the application is the desired action. The Close button mirrors the action of the Windows close button in the upper right corner of the application.

?: This button is generally hidden until a system error is thrown in the application. If a user error is thrown, the user is told exactly why the issue occurred. System errors are internal function errors and usually do not contain information that the user would understand. In this case, when a system error is thrown, the "?" button appears and allows the user to generate the filename and line number of the error. This

information can be used to determine exactly where the problem occurred. Once any part of the GUI is run, the "?" button is hidden again.

Static Textboxes:

Status/Error: The Status and Error static textbox displays messages useful to the user during the processing of data. Status messages are displayed using black text. If an error is thrown during the operation of the GUI, the error message is displayed in the Status and Error static text box using red text. If a user entered value is outside of the acceptable parameters, a detailed message is displayed for the user and the improper value is highlighted in orange so the user can quickly find the incorrect value. If an internal function error is thrown because of incorrect data or incorrect processing, the error is displayed as a system error.

APPENDIX B

GLYCOANALYZER GLOBAL VARIABLE DESCRIPTIONS

This section details every global variable used in the GlycoAnalyzer GUI application. Global variable values are stored in the XLS-file, GlobalVariables.xls, and are loaded into the application upon start-up with a call to the function, Get_globals_GUI.m. Changing the values in GlobalVariables.m will change the values that are loaded into the application the next time it is launched.

Overall Global Variables:

cohort: Contains the name of the cohort or assay.

GID: Array of glycan identification numbers. The values for this variable are loaded directly from the study's data file.

GUI_flag: Details if a function is being called from the GlycoAnalyzer application or on its own in the MATLAB Command Window. If GUI_flag=1, the application is calling the function. If GUI_flag=0, the function is being called outside of the application.

hFig_main: Used to store all data and GUI component handle values required for the operation of the Main window. This value is created when the GUI is first opened by the user.

hFig_output: Used to store all data and GUI component handle values required for the operation of the Output window. This value is created when the GUI is first opened by the user.

hFig_plot: Used to store all data and GUI component handle values required for the operation of the Plot window. This value is created when the GUI is first opened by the user.

LID: Cell array of disease categories considered for the study.

PID: Array of patient identification numbers for the training dataset. The values for this variable are loaded directly from the study's data file.

PIDv: Array of patient identification numbers for the validation dataset. This variable is only populated if there is a validation dataset. Otherwise, PIDv is initialized to the empty set. The values for this variable are loaded directly from the study's data file.

Preprocessing Global Variables -

correlation_flag: Determines if the correlated glycans are combined. If correlation_flag=0, the intensities of the correlated glycans are not combined. If correlation_flag=1, the intensities of the correlated glycans are combined and all correlated glycans that are not combined are removed.

Feature Selection and Projection Global Variables:

sn_desired: Sets the desired sensitivity.

sp_desired: Sets the desired specificity.

Plotting Global Variables:

aspect: Parameter used in the weighting function needed for the calculation of the ROC curve. If aspect=0, no weighting is used for the overall AUC. If aspect=1, AUC is calculated for high specificity. If aspect=2, AUC is calculated for high sensitivity.

bwidth: Parameter used in the second of the two ImmunoRuler plots and determines the width of the plot of the test sample. bwidth is used for the parameter, width, in the function, `bar()`. The standard width of a bar in the MATLAB bar graph is 0.8. If a width of 1 is specified, the bars in the bar graph touch each other with no separation. The standard value for bwidth is 2, meaning the width of the test sample is wider and overlaps the adjacent bars in the ImmunoRuler plot [MATLAB Help Files, search the bar function].

cflag: Parameter used in the second of the two ImmunoRuler plots and determines if an equal cost cutoff line is displayed in the plot. If cflag=0, an equal cost cutoff line is not displayed. If cflag=1, an equal cost cutoff line is displayed.

eflag: Parameter used in the second of the two ImmunoRuler plots and determines if bar edges are displayed in the plot. If eflag=0, bar edges are not displayed in the plot. If eflag=1, bar edges are displayed in the plot.

lflag: Parameter used in the second of the two ImmunoRuler plots and determines if a legend is displayed for the plot. If lflag=0, a legend is not displayed. If lflag=1, a legend is displayed.

ns: Used to remove outliers during calculation of the ROC curve. If ns is specified, data is removed if it is ns standard deviations away from the mean. The outliers are not removed if ns is not specified or ns=0.

pflag: Used during the calculations required for the ImmunoRuler plot and toggles if goodness of training is calculated or not. If pflag=0, execution of the ImmunoRuler plot is faster and goodness of training is not calculated. If pflag=1, goodness of training is calculated.

qflag: Parameter used in the second of the two ImmunoRuler plots and determines how many colors are used in each sample during plotting. If qflag=0, each sample is represented by one color. If qflag=1, two colors are used for each sample.

Wa: Parameter used in the weighting function needed for the calculation of the ROC curve. Wa defines the range in the array of false positive rates.

Wb: Parameter used in the weighting function needed for the calculation of the ROC curve. Wb defines the slope of the weighting function.

Wb: Parameter used in the weighting function needed for the calculation of the ROC curve. Wb defines the slope of the weighting function.

wflag: Parameter used in the second of the two ImmunoRuler plots and determines if whiskers are displayed in the plot of the test sample. If wflag=0, whiskers are not displayed in the plot of the test sample. If wflag=1, whiskers are displayed in the plot of the test sample.

APPENDIX C

GLYCOANALYZER FILES AND FUNCTIONS

This section lists every file used by the application. Each of these files must be specified in the project file used by the MATLAB deploytool during the compilation and packaging processes.

File Name	File Description
analysisErrorChecks_GUI	Checks all data analysis values to make sure they are valid for processing. This includes the proper loading of the training file and all editable textboxes.
axesSelectPDFMain_GUI	Allows user to select on of the smaller PDF plots in the Main window and blow it up into a larger figure window.
axesSelectPDFPlot_GUI	Allows user to select on of the smaller PDF plots in the Plot window and blow it up into a larger figure window.
axesSelectROCMain_GUI	Allows user to select on of the smaller ROC plots in the Main window and blow it up into a larger figure window.
axesSelectROCPlot_GUI	Allows user to select on of the smaller ROC plots in the Plot window and blow it up into a larger figure window.
checkboxErrorChecks_GUI	Checks all the checkboxes to make sure that two of the same cancer types aren't checked at the same time for the Control/Case columns. This function also makes the user select at least one Cancer in each of the Control/Case columns.
clearPlotTextboxes_GUI	Clears all text from the ten training and validation text boxes under the plot axes. This is just a helper function to reduce code in Immunoruler_GUI.
closeOutput_GUI	Hides Output GUI window when user clicks the Microsoft Windows Close button.
closePlot_GUI	Hides Plot GUI window when user clicks the Microsoft Windows Close button.
costErrorChecks_GUI	Checks numerical value of Cost to make sure it is a valid value.
createPrintFile_GUI	Allows the user to print all values that are valid for the current test configuration and the values displayed in the Output window.
dataFileErrorChecks_GUI	Checks to make sure a proper training data file is loaded.
disableButtons_GUI	Disables all buttons, textboxes, and pulldown menus. It also sets the icon of the pointer to that of a watch to show the user the program is working and is busy.
displayFSProjectionOutput_GUI	Displays all feature selection/projection outputs properly in the Output window.

File Name	File Description
displayPreprocessingOutput_GUI	Displays all preprocessing outputs properly in the Preprocessing Output window. These values are found in the file Prepare.m.
enableButtons_GUI	Enables all buttons, textboxes, and pulldown menus. It also sets the pointer to that of a pointer to show the user the program is not busy.
extractCancers_GUI	Extracts an array of which checkboxes are selected in the Control/Case/Test columns in the analysis section.
extractCombineData_GUI	Takes raw, normalized training data and extracts and combines data based on the selected checkboxes in the Control/Case columns of the Control/Case/Test section. This function also extracts the classes from the GUI in the proper order.
Get_globals_GUI	This function retrieves parameters from an XLS file which contains all global paramneters necessary to run the GlycoAnalyzer.
getAnalysisValues_GUI	Gets the Feature Selection and Projection values from the popup menus and editable textboxes in the GlycoAnalyzer.
getPreprocessingValues_GUI	Gets the Plotting values from the popup menus and editable textboxes in the GlycoAnalyzer.
getTestCheckboxes_GUI	Checks if any of the checkboxes are selected in the Test column of the Control/Case/Test section.
getValues_GUI	This helper function gets all the uicontrol values for textboxes, checkboxes, checkbox visibility, editable textboxes and pulldownmenus. This function is used when the user either quits the program or decides to save the GUI uicontrol values.
Immunoruler_GUI	Main file for the GlycoAnalyzer. This file controls the opening and closing of the application as well as the function of all GlycoAnalyzer controls.
largeAxesOff_GUI	Makes the large axes invisible to user in both the Main and Plot windows. This is only for the older version of ImmunoRuler.
largeAxesOffNew_GUI	Makes the large axes invisible to user in both the Main and Plot windows. This is only for the new version of ImmunoRuler.
largeAxesOn_GUI	Makes the large axes visible to user in both the Main and Plot windows. This is only for the old version of ImmunoRuler.
largeAxesOnNew_GUI	Makes the large axes invisible to user in both the Main and Plot windows. This is only for the new version of ImmunoRuler.

File Name	File Description
makeCheckboxesVisible_GUI	Takes in the number of cancers and makes the Control/Case/Test checkboxes visible based on the number of cancers in the LID string.
makePlotTextboxesInvisible_GUI	Makes all textboxes from the ten training and validation text boxes under the Plot axes invisible.
makePlotTextboxesVisible_GUI	Makes all textboxes from the ten training and validation text boxes under the Plot axes visible.